Department of Electrical and Information Engineering

Electrical and Information Engineering

Ph.D. Program

SSD: ING-INF/05-INFORMATION PROCESSING SYSTEMS

Final Dissertation

Knowledge-Enabled Recommender Systems in the Linked Data Era

---

by

Vito Walter Anelli

Referees:

Prof. Alejandro Bellogín

Prof. Mathieu D'Aquin

Supervisors:

Prof. Eng. Tommaso Di Noia

Prof. Eng. Eugenio Di Sciascio

Coordinator of Ph.D. Program:
Prof. Eng. Alfredo Grieco

Course n°32, 01/11/2016-31/10/2019

If your daily life
seems poor,
do not blame it.

Blame yourself,
tell yourself that
you are not poet enough
to call forth its riches.

R. M. R.

Contents

Abstract

Semantic Web, Knowledge Graphs, and Recommender Systems are unfamiliar to ordinary people. However, they are almost everywhere. For instance, in a world that overwhelms us with relevant and irrelevant information, Recommender Systems silently make the difference. They process catalogs from thousands to millions of items to return us only the relevant and personalized information. Otherwise, we would be castaways in the ocean of information that try to drink it all. On the other side, they are constantly whispering in our ear, suggesting to enjoy news, movies, songs. Choosing whether they are Jiminy or Lamp-Wick is our responsibility. At the same time, the Web is evolving, providing us rich and semantic information. The emergence of Knowledge Graphs and the so-called Semantic Web can make Recommender Systems evolve to handle high-quality knowledge. This knowledge would enable Recommender Systems to understand the domain, provide explanations, and improve recommendations. In this research journey, we have faced different aspects of the recommendation and how semantic knowledge can dramatically impact them. We have disassembled and reassembled several Machine Learning algorithms to highly integrate semantics into the recommendation approaches. Feature Factorization, Graph Spreading Relevance, and interpretable Factorization Machines are a few examples. Finally, without limiting ourselves to Machine Learning, we have designed a SPARQL-based recommendation system that uses conditional pair-wise preferences theory, thus automating the understanding of complex human mental processes.

We hope you will enjoy the journey.

Chapter 1

Introduction

In the last decades, we have observed a quick growth of the Web. In the beginning, it was often used only by insiders. Now, it has become a powerful tool used by everyone. Concurrently with its spread, also the role of users has changed. Before, they were only consumers of the available information. Nowadays, they also produce a large amount of information available on the Web. This evolution, named ”Web 2.0” consisted of a mass injection of documents and websites on the Web.

When we type a single keyword on a search engine, it could even return millions of documents. Our parents and grandparents have left us many different senses for the term Freedom, the possibility of choosing among alternatives. However, when we have a million alternatives, a human is not able to process them. In practical terms, having one million alternatives is like having no one. This effect takes the name of ”misery-inducing tyranny”.

To avoid this problem, a new need has emerged of processing, annotating, and indexing these documents to make their search easier. The development of these techniques has led to the search engines era. These agents can process a considerable amount of data to return a relevance value for each item. All the issues related to the search have given birth to a new research field: Information Retrieval. However, only a small fraction of search engines consider some degrees of personalization of search results. With the explicit aim of personalization, a new class of agents has emerged: Recommender systems. The rationale of these tools is filtering out irrelevant information on a per user basis. They are designed to analyze big collections of objects, products, or services and extract personalized relevant shortlists.

To make a Recommender Systems work properly, the system must understand the domain knowledge. Frequently, the problem is worked around by the designers encoding the domain knowledge in the Recommender Systems implementation. However, it is reasonable that all the needed knowledge is already available on the Web. This claim leads directly to the second problem. Until now, we have discussed information and documents on the Web irrespective of their nature. If we consider the Web as a collection of documents, we are claiming that some of these documents contain the desired knowledge. Here, a problem arises. Usually, the interpretation of a non-structured document (i.e., plain text) is a typically human task. Unfortunately, since we have observed that the number of documents on the Web is overwhelming, we obtain a new overloading problem.

This is basically due to the documental nature of the Web, in which documents enclose non-structured information. However, even though each document contained only one structured fact, human interpretation would be unfeasible. These considerations lead to two new needs. The first is these facts should be machine-readable. If we had these kinds of data, a software agent could automatically analyze them. The second is the meaning of data should be explicitly defined. If the semantics comes with data, the software agent can understand it. This new Web is named "Semantic Web".

1.1 Motivation

The emerging of the new knowledge bases based on Semantic Web technologies enabled the creation of a new class of Recommender Systems. The exploitation of this kind of knowledge can improve Recommender Systems under many different aspects and alleviate some known issues. For instance, some traditional Recommender Systems approaches suffer the Cold-Start and the over-specialization problem. Cold-Start is a problem that a designer has to face when a new item or a new user enters the platform. If the suggestions provided to the user are produced based on the platform's past transactions, a new item will not ever be recommended. In this case, we are talking about the Cold-Start item problem. Analogously, if the suggestions are produced based on the user's past transactions, we can not provide any recommendations to the new user. Instead, the over-specialization problem is the tendency of some Recommender Systems to suggest always items that are very similar to the user's history.

These problems indicate that there are many different dimensions to consider to evaluate Recommender Systems. Beyond the classic accuracy dimension, nowadays, we usually also evaluate Coverage, Novelty, Diversity, and Serendipity of results. This research work was born with the idea of exploiting the knowledge enclosed in Linked Open Data datasets to alleviate the Recommender Systems problems.

Almost immediately, we have faced many other problems to inject this knowledge into Recommender Systems. From the choice of the preferred knowledge source to the evaluation of the knowledge source quality, this research work has become a journey through the grey areas of the Semantic Web and the Recommender Systems.

Despite the newborn research field, we have not been alone during this travel. Other researches have perceived the importance of leveraging Linked Data knowledge in Recommender Systems and Information Retrieval. Several technical tracks on Knowledge-based Recommender Systems have begun appearing in top-notch conferences. Although each researcher mainly focuses on a few specific topics, a community has begun to take shape.

1.2 Research Questions

In this dissertation, we have focused on some specific aspects of knowledge-aware Recommender Systems. We have explored whether (and how) the exploitation of semantics in these aspects could be beneficial. In detail, we have focused on a peculiar kind of side information: structured information.

We have explored the possibility of using this information to deal with sparse matrices. Our idea has been to exploit the graph-based nature of Linked Data to compensate for the lack of connections in the original users-items matrix. These matrices frequently have a density of less than 0.1%. If the representation of user preferences moved from items to semantic features, we could exploit the information delivered by a knowledge graph. Another common problem generated by the lack of data is the Cold-Start problem. Again, we have found interesting exploring if structured metadata can help to produce meaningful recommendations even in extreme scenarios like Cross-Domain recommendations. Another common problem we have wanted to face is the interpretability of the recommendations. The most effective models exploit latent factors that can not be associated with any explicit features. For this reason, interpret a particular recommendation list is not possible. Eventually, this leads to a gradual decrease in trust. We have tried to adapt Latent Factors models to work with Linked Data knowledge. We have substituted latent factors with explicit knowledge to make these models interpretable. This has been a particularly exciting research line. The most challenging aspect has been measuring if the system was preserving the original semantics.

Based on these considerations, we have formulated the following research questions:

- RQ1 – Can the injection of Linked Data alleviate the sparsity problem in Collaborative-Filtering Recommender systems?

- RQ2 – Can we exploit Linked Data to face the Cold-Start Problem?

- RQ3 – Is semantic knowledge a way to make Recommender Systems interpretable?

- RQ4 – Can we define a recommendation approach based on pair-wise preferences elicitation?

- RQ5 – Can an Information Retrieval weighting scheme also consider collaborative information?

Almost all the ideas proposed in this research work have passed a peer-review, and they are published in Specialistic Journals or International Conferences Proceedings. To provide a more holistic overview, we have reported the contributions connecting them to the related research questions:

- RQ1 – Can the injection of Linked Data alleviate the sparsity problem in Collaborative-Filtering Recommender systems?
  - **Feature Factorization for Top-N Recommendation: From Item Rating to Features Relevance,** [18] RecSysKTL @RecSys 2017 (CORE: A)
  - **Feature Augmentation in Top-n Recommendation Scenarios via Linked Data,** Original research, not published yet
- RQ2 – Can we exploit Linked Data to face the Cold-Start Problem?
  - **Addressing the user cold start with cross-domain collaborative filtering: exploiting item metadata in matrix factorization,** [95] User Modeling and User-Adapted Interaction Journal 2019 (SJR: Q1)
- RQ3 – Is semantic knowledge a way to make Recommender Systems interpretable?
  - **How to make latent factors interpretable by feeding Factorization machines with knowledge graphs,** [21] ISWC 2019 (CORE: A). **This work has been awarded as Best Student Research Paper.**
  - **Semantic Interpretation of Top-N Recommendations,** [15] IEEE TKDE 2020 (SJR: Q1)
  - **Knowledge-Aware Interpretable Recommender Systems** [12] Book Chapter
- RQ4 – Can we define a recommendation approach based on pair-wise preferences elicitation?
  - **Combining RDF and SPARQL with CP-theories to reason about preferences,** [273] Semantic Web Journal 2019 (SJR: Q1)

- RQ5 – Can an Information Retrieval weighting scheme also consider collaborative information?
  - **A Principled Approach to Hybrid Relevance in Top-N Recommendation.** Original research, not published yet

As a careful reader can imagine, this research work also covers some other topics like Semantic Knowledge extraction and Semantic Data management. For the sake of space, and to avoid dispersive storytelling, we have excluded them from this research summary. To provide a reader with a more comprehensive overview, we report below the reference to these works:

- Semantic Knowledge extraction
  - **LOSM: a SPARQL endpoint to query Open Street Map,** [187] 14th International Semantic Web Conference, ISWC 2015 (CORE: A)
  - **Exposing Open Street Map in the Linked Data cloud,** [13] IEA/AIE 2016 (CORE: C): The 29th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, 2016
  - **Querying deep web data sources as linked data.** [10] WIMS 2017
- Semantic knowledge management
  - **Etytree: A Graphical and Interactive Etymology Dictionary based on Wiktionary.** [203] WWW (Companion Volume) 2017 (CORE: A*)

Additionally, we have organized two workshops in top-notch conferences, to connect researchers with the same research interests:

- **Knowledge-aware and Conversational Recommender Systems Workshop**, [9, 17] RecSys 2018 (CORE: A)
- **Second Knowledge-aware and Conversational Recommender Systems Workshop**, [16] CIKM 2019 (CORE: A)

Finally, beyond the exact scope of the Ph.D. research topic, we also focused on semi-structured information and substantially contributed with some advacements to the Recommender Systems research field. In detail, we tried to answer the following questions:

- Is that possible to attack (and protect) Recommender Systems using semantic knowledge?
  - **SAShA: Semantic-Aware Shilling Attacks on Recommender Systems Exploiting Knowledge Graphs,** [14] ESWC 2020 (CORE: A)
- Can we formulate a diversification scheme that takes into account temporal aspects?
  - **An Analysis on Time- and Session-aware Diversification in Recommender Systems.** [11] UMAP 2017 (CORE: B)
- Is that possible to formulate a personalized notion of the Popularity of items that also considers Time?
  - **Time-aware Personalized Popularity in top-N Recommendation,** [20] ComplexRec @RecSys 2018 (CORE: A)
  - **Local Popularity and Time in top-N Recommendation, 41st European Conference on Information Retrieval,** [23] ECIR 2019 (CORE: A)
- What happens in Recommender Systems performance if we exploit a more complex notion of similarity that also considers dissimilarity and asymmetric similarity?
  - **The Importance of being Dissimilar in Recommendation,** [22] 34rd ACM/SIGAPP Symposium On Applied Computing, SAC 2019 (CORE: B)

- Are the evaluation metrics equally valid for tuning Recommender Systems hyperparameters? Dually, do the different parameters have the same degree of importance?
  - **On the discriminative power of Hyperparameters in Cross-Validation and how to choose them,** [19] RecSys 2019 (CORE: A)
- Is that possible measuring a generalized notion of fairness exploiting semi-structured information?
  - **A Flexible Framework for Evaluating User and Item Fairness in Recommender Systems,** [75] UMUAI 2021 (SJR: Q1)

1.3 Structure of the work

The remainder of this dissertation is structured as follows. Part I is devoted to providing a general overview of the necessary technologies useful to understand the more advanced approaches presented in the following parts. We have first given an idea of what Semantic Web is. We start from a historical viewpoint to motivate the necessary advent of Semantic Web. Then, we briefly describe the technologies we widely use in this work. After these sections, we provide a brief introduction of preference representation.

The following chapter focuses on Recommender Systems. An introduction to the research field opens the chapter. Next, we describe some of the most known approaches for the recommendation. Sections devoted to optimization techniques and evaluation metrics close the chapter.

The following part focuses on the main research line of this work. Part II investigates the exploitation of explicitly formalized knowledge. This part covers different research lines from the leveraging of semantic features in matrix factorization to model interpretability, to the generation of recommendations based on pair-wise preferences.

Part III is devoted to drawing some conclusions about this research work.

Part I

Background

Chapter 2

Knowledge Representation

This section is devoted to a brief overview of the basic concepts regarding knowledge representation. In detail, the chapter describes the birth and development of the Semantic Web and the Linked Open Data initiative. In particular, the aim is to provide the crucial ideas and technologies that stand behind the broad term of the Semantic Web.

We introduce to the staple language, Resource Description Framework (RDF). Then we briefly describe the query language SPARQL. The following chapter focuses on the representation of Preferences. We motivate its importance in this work, and we provide a brief overview of the ”ceteris paribus” theories.

This section covers only the most basic concepts about them. A more exhaustive literature review about advanced theories and applications is realized on a per research line basis in the following sections. This choice was made necessary because the corresponding literature reviews could seem a bit confusing if the reader is not already involved in the specific research field. We hope this choice lets the reader appreciate these fascinating concepts properly.

2.1 Semantic Web

The extensive and pervasive growth of the Web has produced the most voluminous information archive ever. The amount of available information is so large that there is no chance that a single person would be able to analyze and index it. The reason for it is also due to the nature itself of the Web, which is document-based and fragmented. Most of the available information is unstructured. It misses the optional additional data necessary to analyze it before accessing the document itself. Moreover, entire sections of the Web are, de facto, inaccessible due to linguistic barriers. These are only some examples of the necessity of a total rebuilding (and rethinking) of the Web. It was needed to move the focus of the Web from documents to the data. The single facts, the assertions, the data should be the heart element of the Web instead of documents. It is straightforward that these data should have a unique interpretation. These data should not be ambiguous as the natural language usually is. Even though we would have built this so-called Web of Data, a single person still would not be able to analyze the whole Web. This consideration has two consequences. The first one is the awareness that the Web can be analyzed only through an automatic process that excludes the necessity of human interpretation of data. Hence, data should be machine-interpretable. The second consequence is that the automatic interpretation is only possible if data comes with an explicit semantics attached to it. This new Web, composed of interpretable data, is called Semantic Web. The birth of the Semantic Web is commonly associated with the publication of an article by Tim Berners-Lee on Scientific American in 2001 [38]. The article is focused on what the Web could become, and in particular, a sentence could serve as an ”Ante-literam” definition: ”The Semantic Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation”. Hence, the Semantic Web is an information network that lets machines understand the contained information. Human users can thus prepare complex queries that machines can understand and process. The underlying needed comprehension tacitly imposes to exploit a well-formalized knowledge representation. The knowledge representation would have been just the first step in the process of development of the Semantic Web. Although, since the organization of knowledge and its interpretability is crucial, it has quickly become the keystone of the development of the Semantic Web. This branch of the Semantic Web, specifically devoted to the representation and organization of knowledge archives (datasets), has obtained a specific name, Linked Data. In 2006, Berners-Lee published a new article [241] partially rethinking the idea of the Semantic Web under a Linked Data perspective. The community has spent a big effort to define and refine the languages and technologies involved. Nowadays, these technologies are exploited to expose data on the Web and connect data. This has resulted in a brief but summarizing assertion: "Linked Data is the Semantic Web did well".

2.1.1 Resource Description Framework

Resource Description Framework (RDF) is a framework designed to represent knowledge. It is one of the pillars of Semantic Web, proposed by the World Wide Web Consortium (W3C) in 1998. The knowledge encoded in RDF can be easily be represented in the form of a graph, queried, and processed by automatic reasoning techniques. Along with the framework, different serialization formats have been proposed, that avoid redundancy and allow wide interoperability. RDF is a general model useful to describe the resources on the Web. The skeleton of the RDF syntax, its Data model, is a set of triples. Each of them is composed of a subject, a predicate, and an object. An RDF triple could express an assertion, a logic expression, or a generic statement. An example of an RDF triple is:

```
<http://example.org/#spiderman>
<http://www.perceive.net/schemas/relationship/enemyOf>
<http://example.org/#green-goblin>.
```

A set of RDF triples is named RDF graph. An RDF graph is a directed graph, in which each triple is a connection Vertex-Arc-Vertex. Formally, an RDF graph is the conjunctive set of its triples. A triple is composed of three distinct elements: the subject expressed as Internationalized Resource Identifier (IRI) or as a Blank Node; the predicate expressed as IRI; the object expressed as IRI, as a Literal, or as a Blank Node. Different triples can share the same IRI. IRIs, Literals, and Blank Nodes since they compose RDF triples are called RDF terms.

IRIs [87] are a generalization of Uniform Resource Identifiers (URIs) [37] that enable the usage of a more wide range of Unicode characters. This means that all URIs can be considered IRIs, but the vice-versa is not necessarily true. The definition and the meaning of IRIs are apart from RDF. On the one hand, this implies absolute freedom in the IRIs creation process. On the other hand, the same IRIs can be used by third-party organizations in their knowledge bases.

Literals are used to express values like strings, numbers, and dates. Moreover, in the case of strings of type `http://www.w3.org/1999/02/22-rdf-syntax-ns#langString`, it is allowed to add a language tag. Literals are composed of three optional elements: value, type, and language. For this reason, two Literals are considered equals only if the three elements are equals. Datatypes are used with Literals to represent the different kinds of data. The adopted abstraction in RDF is XML Schema compatible. Thus, it is possible to introduce new custom data types and use them in RDF. A data type is composed of: lexical space, a value space, and a lexical-to-value mapping. The lexical space is a set of unicode strings. The lexical-to-value mapping represents a function which has as domain the lexical space and as range the value space.

Blank Nodes are local identifiers that denote a generic RDF term that is present within the considered collection, with a local scope. They usually used if we are not interested in the values of the variable associated with the Blank Node.

Resources, or entities, can be defined using IRIs or strings. Resources can indicate everything: physical objects, documents, concepts, numbers, strings. An example of the description of the ”Pulp Fiction” RDF resource is:

```
<http://dbpedia.org/resource/Pulp_Fiction>
<http://www.w3.org/2000/01/rdf-schema#label>
"Pulp Fiction"@en
<http://dbpedia.org/resource/Pulp_Fiction>
<http://dbpedia.org/ontology/director>
```

`<http://dbpedia.org/resource/Quentin_Tarantino>`

which states that the resource has an English name, "Pulp Fiction", and the resource has been directed by Quentin Tarantino.

RDF vocabulary is a collection of IRIs that can be used in an RDF graph. RDF schema, for instance, is an RDF vocabulary. Moreover, RDF vocabularies can be further used to define other vocabularies. If the initial part of the path of the IRIs is common among more IRIs, this substring is usually defined as a namespace. These namespaces can also be associated with a shorter and intuitive string named prefix. Some RDF serialization formats exploit prefixes to reduce the verbosity of IRIs.

RDF Dataset is a collection of RDF graphs. RDF Datasets have some specific features: i) one RDF graph has to be defined as default graph; ii) they contain zero or more named graph. A named graph is a pair composed of an IRI or a Blank Node, and an RDF graph. To conclude this brief overview of RDF, we mention **rdf:type** predicate (`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`). This predicate is particularly important under an ontological perspective since it states the belonging of the subject to a certain object class.

2.1.2 RDF Schema

RDF model is a flexible graph-based model that can be easily used for knowledge representation. Nevertheless, it contains only a few raw tools to define the semantics of the resources. For this reason, RDF Schema (RDFS), has been proposed. RDFS provides essential tools to define ontologies. Ontologies, or RDF vocabularies, are a useful tool to structure domain knowledge. RDFS eases the definition of classes and their properties. It provides ways to define which properties should be used with instances of specific classes. RDFS is completely compatible with RDF. Thus an RDFS vocabulary still is a valid RDF graph. In particular, RDFS is formalized as an RDF vocabulary with the namespace `www.w3.org/2000/01/rdf-schema#` (with the common prefix **rdfs:**). To define the kind of resource, `RDFS` provides the term *Class*. Classes are described by the `RDFS` resources **`rdfs:Class`** and

rdfs:Resource, **rdfs:subClassOf**, and through the well-known `RDF IRI` **rdf:type**. As the reader could imagine, these entities let create a hierarchical organization of the entities described in the RDF graph. A single entity can still be defined as belonging to more than one class. The predicate **rdfs:domain** and **rdfs:range** are intended to describe other properties. In detail, **rdfs:domain** denotes that the subject of the triple that makes use of the target property should belong to the class indicated in the object. **rdfs:range** denotes that the object of the triple that makes use of the target property should belong to the class indicated in the object. **rdfs:subClassOf** is a particular predicate with domain and range entities of the type rdfs:Class. Furthermore, the properties can be hierarchically organized, making use of **rdfs:subPropertyOf**. Finally, RDFS provides rdfs:comment and rdfs:label that lets insert comments and labels in natural language.

2.1.3 Linked Data

One of the core points of the documental Web was to connect different documents through specific links, hyperlinks. Similarly, in the Semantic Web, the connections between data are fundamental. The datasets that provide links with other datasets and the technologies involved are consequently named Linked Data. To enable modern applications to work with these knowledge bases, data should be exposed using standard formats, should provide an endpoint to query the knowledge base. When a dataset also provides connections with other datasets, we can define it as a Linked Dataset. Two common examples of Linked Datasets are DBpedia and WikiData, which basically expose the content of Wikipedia pages following the Linked Data best practices. Since DBpedia has been used extensively during this thesis, I have chosen it as an example in the following paragraphs. I hope this will make the reader more familiar with this dataset, enabling her to understand the remainder of the thesis better. To better explain the importance of inter-datasets links, let us consider an example. DBpedia provides, for instance, links to the Geonames dataset. Providing this information, expressed as triples, enables the creation of new applications that can access the information about a resource, retrieve the Geonames link, access Geonames, and retrieve additional information. It is important to underline that all the additional information that is present in Geonames is not present in DBpedia. These connections let developers build applications that can integrate different data sources easily. The Linking Data Initiative started in 2007 intending to extend the actual Web using the standards proposed in the Semantic Web. In this while, a huge number of datasets have been created and connected together, composing the so-called Linked Open Data (LOD) cloud. The LOD cloud clearly shows that this new Web is not substituting the documental Web, but it is enriching it.

2.1.4 SPARQL

SPARQL (Simple Protocol and RDF Query Language) is the de-facto standard query language for RDF data. The language was formalized by Data Access Working Group of W3C (World Wide Web Consortium) in 2008. A newer formalization, SPARQL 1.1, was released in 2013. SPARQL 1.1 shares most of the RDF specifics, but it is stricter in some cases. An example is the identification of an RDF graph that, in SPARQL 1.1, does not allow the usage of Blank Nodes. SPARQL is a key element of the Semantic Web and it lets extract information from remote RDF knowledge bases. The fundamental role of SPARQL is really simple: it processes the RDF graph finding one or more specific subgraphs that correspond to the informative need expressed by the query. The SPARQL processing is made possible by providing two graphs: the data graph (usually already loaded in a remote RDF store) and the query graph (even here described through triples). SPARQL adopts the Turtle syntax, an intuitive and non-verbose RDF serialization format.

Assertions are expressed employing subject-predicate-object sequences, and they are terminated by a full stop. PREFIX introduces prefixes and namespaces, whereas URIs are enclosed within angle brackets and Literals within quotation marks. With only these elements we can already provide an example:

```
@prefix cd: <http://example.org/cd/>
@prefix: <http://example.org/esempio/>
```

```
:Permutation cd:autore "Amon Tobin".
:Bricolage cd:autore "Amon Tobin".
:Amber cd:autore "Autechre".
:Amber cd:anno 1994.
```

The results of the SPARQL query can be returned in more or less machine-understandable formats: XML, JSON, RDF, HTML.

The query language reflects the graph-based nature of the data model. Indeed, the query-answering mechanism is done via pattern-matching over the knowledge graph (graph-matching). Usually, a SPARQL query is composed of at least one graph pattern. A graph pattern is a set of triples, also called triple-pattern. These patterns are similar to RDF, thus they consist of a subject-predicate-object sequence. The main difference is that each of these components could be a variable. The query is solved by looking for the subgraphs that correspond to the pattern. In particular, the search for the subgraphs aims to find all the possible instantiations of RDF terms with variables that correspond to existing triples in the knowledge graph. The basic syntax is inspired by SQL. For instance, SPARQL uses the SELECT clause to define the variables that will appear in the results, whereas the WHERE clause encloses the graph pattern. As ane example, a graph pattern can be: ?title cd:author ?author . where the two variables are denoted by the question mark. Similarly to SQL, we can exploit the GROUP BY clause to aggregate results or FILTER to filter results based on some conditions. Instead of SELECT, we can also use other clauses as ASK, which returns a boolean value, DESCRIBE, which returns the complete description of a resource, and CONSTRUCT, which returns graphs.

2.2 Preference representation

Dealing with user preferences is an important aspect of every application designed to provide personalized information to the end-user. The original interest in preferences can be found in decision theory, as a way to support complex, multifactorial decision processes [98], and nowadays every personalized system needs a preference model to capture what the user likes or dislikes. Once the user model has been represented, it is then exploited to filter information coming from a data source, e.g., a database, in order to provide a ranked list of results matching the order encoded in the preferences of the user.

Query languages usually let us specify the information that we want to be returned (hard constraints), where the result set contains elements with no specific order with reference to user preferences. It contains all those resources that exactly match the constraints represented by the query. As a matter of fact, if just one of the requirements representing the query is not fulfilled, the result set can be empty. At the same time, returning huge and unordered sets of answers could be useless and even counter-productive. A possible way to bypass these issues is to allow the language to represent both hard constraints—used to return only relevant results—and soft ones, i.e., preferences—to rank the results by fulfilling a user's tastes. Approaches to preference representation can be either quantitative or qualitative [78]. The formers are based on a total ordering of the outcomes, given by a scoring function, while the latters enable the representation of partial orders since preferences are treated as independent dimensions. From a user perspective, a qualitative approach is more natural than the quantitative one [195]. Indeed, in the first case, the user has just to provide pairwise qualitative comparisons, while in the second case, she has to assign a value to many alternatives, which very often are represented in a multi-attribute setting. Regarding the Linked (Open) Data world, the notion of qualitative preferences in `SPARQL` queries was introduced in [248], by Siberski et al., whose *preference-based querying* language extends `SPARQL` through the introduction of solution modifiers (the `PREFERRING` clause). Their query formulation retrieves only items that are the most preferred ones, or equivalently *undominated*. The work [111] builds on the earlier approach of [248], but adds preferences at the level of filters, rather than as a solution modifier. The *PrefSPARQL* syntax of [111] needs no additional solution modifier to express qualitative preferences, as it leverages the expressive power of `SPARQL` 1.1. However, the approaches proposed in [248] and [111] both have an important limitation: they are not able to provide an order of all the available outcomes that reflects user preferences. That is, they return only the undominated (a.k.a. Pareto-optimal) results, i.e., those outcomes best satisfying user preferences. Unfortunately, the size of the resulting answer set could be too small to be of practical use. This fosters moving beyond the Pareto-optimal set identification to a top-k scenario [195], where firstly available outcomes are ordered, even if with the ties implied by a qualitative approach, from best (most preferred) to worst (less preferred) according to a given user's preferences, and then the first k results are returned.

2.2.1 CP-theories

Utility functions can be considered as the ideal tool for representing and reasoning with preferences, but the total order that they allow representing does not always reflect the actual user model. Partial orders among preferences are a more natural way to represent a user's tastes. Qualitative statements, e.g., "*given u, I prefer x_i over x_i'*", permit a system to encode partial orders among user preferences, thus granting the representation of a more realistic user model. Let us consider the following example. The previous example is a typical conditional statement where the core notion of a CP-statement is explicitly represented. We have that when a particular condition u is true, a user prefers to enjoy items where also x_i is true, rather than items where x_i' is true, given that x_i and x_i' cannot be true at the same time.

Example 1 (Books)
"Giorgio has just finished his exams and wants to relax with a book. Giorgio can read both English and French, but he would like to improve his French to enrich his curriculum and so he prefers to read French books. Giorgio prefers reading crime books over autobiographical ones for French books since he believes that crime plots are more captivating and, therefore, more useful while learning a foreign language. The reverse order holds for English books. Giorgio is a good reader and, therefore, given an English book, he prefers those being part of a saga. The literary genre and the presence of a subsequent work not have the same importance to Giorgio: in case of English books, he considers the choice on the genre more important than the one dependant on sequels, while the opposite happens for French books. Finally, for books characterized by a sequel, Giorgio regards the presence of

a cinematographic version positively". ■

By looking at Example 1, we find it quite hard (even impossible) to directly represent Giorgio's preferences by means of a score assigned to his preferential statements. In fact, Giorgio's preferences can be better expressed as qualitative (pairwise) comparisons.

Relevant frameworks to represent and reason with qualitative preferences are built according to the *ceteris paribus* semantics [115] and specifically consist of conditional preference networks (or CP-nets) [44] and a formalism along similar lines to CP-nets, but with a richer language of preference statements, namely, conditional preference theories (or CP-theories) [280].

Syntax. Formally, given a set of variables V, a CP-theory Γ is a set of preference statements φ of the general form:

$$u_\varphi : x_\varphi > x_\varphi^{'} [W_\varphi],$$

where u_φ is an assignment to a set of variables $U_\varphi \subset V$, x_φ and $x_\varphi^{'}$ are different assignments to some variable $X_\varphi \notin U_\varphi$, and W_φ is some subset of $V - U_\varphi - \{X_\varphi\}$.

Semantics. The interpretation of φ is that, given u_φ, x_φ is strictly preferred to $x_\varphi^{'}$, all else being equal, but irrespective of the values of variables in W_φ.

That is, φ compactly states that for all assignments w, $w^{'}$ to W_φ and assignments t to $T_\varphi = V - U_\varphi - \{X_\varphi\} - W_\varphi$, $tu_\varphi x_\varphi w$ is preferred to $tu_\varphi x_\varphi^{'} w^{'}$.

In what follows, we will use the word *outcome* to indicate a complete assignment to all the variables in V and denote the set of all outcomes as $\mathcal{O}$. For the statement φ, we denote by φ^* the set of pairs of outcomes $(tu_\varphi x_\varphi w, tu_\varphi x_\varphi^{'} w^{'})$, where t is an assignment to T_φ, and w, $w^{'}$ are assignments to W_φ. Further defining $\Gamma^* = \cup_{\varphi \in \Gamma} \varphi^*$, it is then natural to define for the CP-theory Γ a strict partial order $>_\Gamma$, induced by Γ on the set of outcomes $\mathcal{O}$, as the transitive closure of Γ^*. The CP-theory formalism allows to express the usual CP-net ceteris paribus statements by simply considering $W_\varphi = \emptyset$ and identifying U_φ with $Pa(X)$, the parents of

φ_1	$\top : C_F > C_{UK}[\emptyset]$
φ_2	$C_{UK} : LG_A > LG_C[SW]$
φ_3	$C_F : SW_{No} > SW_{Yes}[LG]$
φ_4	$C_{UK} : SW_{Yes} > SW_{No}[\emptyset]$
φ_5	$C_F : LG_C > LG_A[\emptyset]$
φ_6	$SW_{Yes} : F_{Yes} > F_{No}[\emptyset]$
φ_7	$SW_{No} : F_{No} > F_{Yes}[\emptyset]$

Table 2.1: The CP-theory $\Gamma_{\mathrm{C-LG-SW-F}}$.

a variable X, that is, variables which the preferences on X depend on. However, as anticipated in Section 8.1, under the stricter CP-net formalism, for each variable X, a parent set $Pa(X)$ must be defined and instantiated when the preference order over values of X is established. This is not required in CP-theories, where you can find more statements related to the same variable X, but with different sets U. In addition, CP-theories allow stronger conditional preference statements than CP-nets, which are natural for users to express. For example, they represent a formalism even more general than TCP-nets [45], an enhancement of CP-nets where (conditional) relative importance between variables can be expressed. TCP-nets can be represented through statements with W containing at most one variable. Moreover, there are statements, such as *I prefer x_i over $x_i^{'}$ irrespective of the values of all other variables*, that cannot be expressed in CP-nets or TCP-nets, but correspond in the new formalism of CP-theories to $\top : x > x^{'} [V - \{X\}]$, where $\top$ is the assignment to an empty set of variables.

Example 2 (Books cont'd)
The overall profile of Giorgio may be modelled by means of the CP-theory $\Gamma_{\mathrm{C-LG-SW-F}}$, that is, the set of statements given in Table 2.1. There, a set of binary variables $V = \{Country, LiteraryGenre, Sub\text{- } sequentWork, FilmVersion\}$ (abbreviated as C, LG, SW, and F, respectively) is considered. Their domains are given by:

- $dom(Country) = \{C_F, C_{UK}\}$ (for *France* and *United Kingdom*),
- $dom(LiteraryGenre) = \{LG_C, LG_A\}$ (for *Cri- me_fiction* and *Autobiographical_novel*),
- $dom(SubsequentWork) = \{SW_{Yes}, SW_{No}\}$ (indicating if a book has a sequel or not),
- $dom(FilmVersion) = \{F_{Yes}, F_{No}\}$ (indicating whether there is a cinematographic version of the book or not). ■

For a CP-theory Γ, the preference ranking over outcomes $>_\Gamma$ introduced above, can be equivalently induced under the *worsening swap* semantics. Hereafter, we use the notation $o(X_i) = x_i$ to indicate that the variable X_i is assigned the value x_i in o, and analogously $o(\{X_j,\ldots,X_{j+k}\}) = \{x_j,\ldots,x_{j+k}\}$ to state that $X_j = x_j,\ldots,X_{j+k} = x_{j+k}$ in o.

Given two outcomes o and $o^{'}$ of $\mathscr{O}$, there is a worsening swap from o to $o^{'}$, if there exist a variable
$X_i \in V - \{X_j,\ldots,X_{j+k}\} - W, x_i, x_i' \in dom(X_i)$
and an assignment $x_j,\ldots,x_{j+k}$ to the variables set $\{X_j,\ldots,X_{j+k}\}$ such that:

(i) $o(X_i) = x_i$ and $o'(X_i) = x_i'$,

(ii) $o(\{X_j,\ldots,X_{j+k}\}) = o'(\{X_j,\ldots,X_{j+k}\}) = \{x_j,\ldots,x_{j+k}\}$,

(iii) $o(V - \{X_i\} - \{X_j,\ldots,X_{j+k}\} - W) = o'(V - \{X_i\} - \{X_j,\ldots,X_{j+k}\} - W)$, and

(iv) $x_j \ldots x_{j+k} : x_i > x_i'[W] \in \Gamma$.

The preference relation $>_\Gamma$ over $\mathscr{O}$ is therefore the transitive closure of worsening swaps.

A CP-theory Γ is *consistent*, if it has a model, i.e., if there exists a strict total order $>$ that *satisfies* Γ, which is equivalent to $> \supseteq \Gamma^*$, that is, $>$ extends $>_\Gamma$. In [279], it is proved that the irreflexivity of $>_\Gamma$ (or equivalently the acyclicity of Γ^*) is a necessary and sufficient condition for consistency.

The consistency of a CP-theory implies that there are no cycles generated by $>_\Gamma$. Avoiding cycles is of paramount importance, as they introduce conflicting information while ordering the outcomes in $\mathcal{O}$. Let us consider the ordering represented in Figure 2.1, where an edge from o_i to o_j represents $o_i >_\Gamma o_j$. As we cannot establish what is the correct ordering of the outcomes due to the cycle created by o_2 and o_3, we could even have situations where we cannot compute the most preferred (undominated) outcome. For example, suppose in Figure 2.1, we do not have o_1. What would then be the best solution for the user in this case?

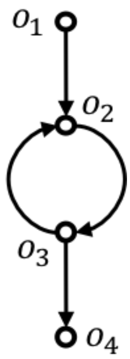


Figure 2.1: An example of cycle among outcomes.

A necessary condition for consistency is local consistency. Consider a CP-theory Γ, a variable $X \in V$, and an assignment a to a set of variables $A \subseteq V$. An ordered pair $(x, x^{'})$ of X values is *validated* by a, if there exists a statement φ of the form $u : x > x^{'} [W]$ in Γ, such that a extends u, that is, a projected to U_φ gives u.

The *local ordering* $\succ_a^X (\Gamma)$ (abbreviated as $\succ_a^X$) on X values is defined as the transitive closure of the set of pairs $(x, x^{'})$ validated by a. Γ is *locally consistent*, if $\succ_\alpha^X$ is irreflexive for all variables X and outcomes α. Local consistency is a necessary condition for consistency, since if Γ is not locally consistent, then there exist an outcome α, a variable X, and a sequence $x_1, \ldots, x_k$ of values of X with associated statements $u_i : x_i > x_{i+1} [W_i] \in \Gamma$ such that α extends u_i and $\alpha(X) = x_1 = x_k$.

This gives a worsening swapping sequence from α to α (only involving changing variable X), thus implying that $>_\Gamma$ is not irreflexive, or equivalently that Γ is not consistent. In general, deciding whether a CP-theory is locally consistent is coNP-complete, but it can be shown that if the size of the parent sets and the size of the domain sets are bounded by a constant, then deciding local consistency is polynomial [280]. Moreover, for CP-nets and TCP-nets, local consistency is always guaranteed [280].

Given a CP-theory Γ, there are several kinds of directed graphs that can be defined on the set of variables V. For $S,T \subset V$, we indicate with $S \rightarrow T$ the set of edges $\{(X,Y) : X \in S, Y \in T\}$, omitting the set brackets, if S or T is a singleton set, e.g., abbreviating $S \rightarrow \{Y\}$ with $S \rightarrow Y$.

The *dependency graph* $H(\Gamma)$ consists of edges $U_\varphi \rightarrow X_\varphi$ for all φ in Γ, that is, all the pairs of the form (Y, X_φ), with $\varphi \in \Gamma$ and $Y \in U_\varphi$. That is, the edge (Y,X) belongs to $H(\Gamma)$ iff there is some conditional preference statement $\varphi \in \Gamma$ that makes the preferences for X conditional on Y. Relative importance is not encoded in $H(\Gamma)$.

On the other side, we define $G(\Gamma)$ to contain $U_\varphi \rightarrow X_\varphi$ and $X_\varphi \rightarrow W_\varphi$ for all φ in Γ, i.e., $G(\Gamma) = H(\Gamma) \cup \{X_\varphi \rightarrow W_\varphi \mid \varphi \in \Gamma\}$. $G(\Gamma)$ contains both dependency and relative importance information: it is $H(\Gamma)$ with the addition of edges (X,Z), if there is any statements φ representing a preference on values of X irrespective of the value of Z (then, X is *more important* than Z, with importance meant as in the TCP-net formalism [45]).

A CP-theory Γ is *fully acyclic*, if $G(\Gamma)$ is acyclic. For fully acyclic CP-theories, consistency and local consistency are equivalent [280].

For a CP-theory Γ and assignment a to a set of variables $A \subseteq V$, we can define another directed graph $J_a(\Gamma)$ on V made of the set of edges $U_\varphi \rightarrow \{X_\varphi\} \cup W_\varphi$ for all $\varphi \in \Gamma$ and also the set $\{X_\varphi\} \rightarrow W_\varphi$ for all $\varphi \in \Gamma$ such that $U_\varphi \subset A$ and a extends u_φ.

A CP-theory Γ is *context-uniformly conditionally acyclic* (or *cuc-acyclic*), if it is locally consistent, and for each outcome $o \in \mathcal{O}$, $J_o(\Gamma)$ is acyclic. It can be proved that a cuc-acyclic CP-theory is always consistent [279]. The condition of cuc-acyclicity is weaker than the full acyclic one, and it requires only $H(\Gamma)$ to be acyclic [280] instead of $G(\Gamma)$.

Cuc-acyclicity implies a less expressive power in terms of preferences that the user may express, but, as we will see in the following, it grants a nicer computational complexity when ranking a set of outcomes (along with guaranteeing always consistency, while deciding the consistency of general CP-nets and CP-theories is PSPACE-complete [108]).

Example 3

To better clarify the expressive limits of cuc-acyclic CP-theories, we consider the following CP-theory $\hat{\Gamma}$ whose $H(\hat{\Gamma})$ is shown in Figure 2.2.

(1) Crime Fiction : Agatha Christie > Andrea Camilleri [$\emptyset$]

(2) Isaac Asimov : Science Fiction > Crime Fiction [$\emptyset$]

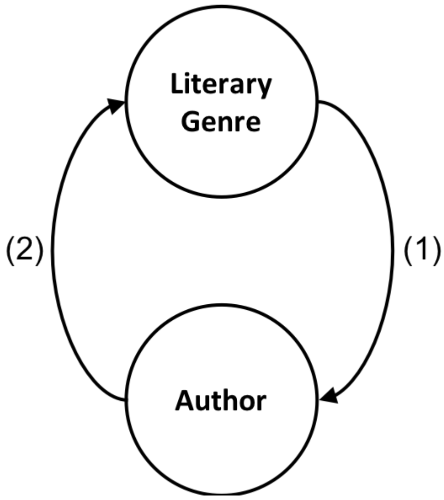


Figure 2.2: The graph $H(\hat{\Gamma})$ representing preferences in Example 3.

Due to the interrelation between the two preferences, $H(\hat{\Gamma})$ is cyclic, and so $\hat{\Gamma}$ cannot be cuc-acyclic.

Although there might be cases where cuc-acyclicity is a strong limitation in the representation of user preferences, it represents a limited subset of all possible CP-theories that can be used to model a user profile. ■

In what follows, we show two results, both proved in [280], which determine a strict partial order extending $>_{\Gamma}$. The approaches proposed to compare outcomes are strongly related to the ordering queries defined in [44] and already discussed in [226] for CP-nets, and can be seen as a generalization of Corollary 4 and Theorem

5 of [44]. The first result is applicable for a fully acyclic CP-theory Γ. It proposes to compare two outcomes by looking (using the appropriate local ordering) at their value on each of the most important variables on which they differ, where importance is defined according to the graph $G(\Gamma)$.

In Theorem 1, we denote with $\Delta(\alpha,\beta)$ the set of variables of V on which outcomes α and β differ, i.e., $\Delta(\alpha,\beta) = \{Y \in V | \alpha(Y) \neq \beta(Y)\}$. If $\alpha \neq \beta$, we build $\Theta(\alpha,\beta)$ as the set of $G^{'}$-maximal elements of $\Delta(\alpha,\beta)$, being $G^{'}$ the transitive closure of $G(\Gamma)$. $\Theta(\alpha,\beta)$ is therefore the set of variables $Y \in \Delta(\alpha,\beta)$ such that there exists no $Z \in \Delta(\alpha,\beta)$ with $(Z,Y) \in G^{'}$.

Theorem 1

Let Γ be a locally consistent and fully acyclic CP-theory, and let the binary relation $\succ_{p(\Gamma)}$ on $\mathscr{O}$ be defined as follows, for any pair of outcomes α and β: $\alpha \succ_{p(\Gamma)} \beta$ iff $\alpha \neq \beta$ and $\alpha(Y) \succ_{\alpha}^{Y} \beta(Y)$ for all $Y \in \Theta(\alpha,\beta)$. Then, $\succ_{p(\Gamma)}$ is a strict partial order extending $>_{\Gamma}$, and the comparison between any pair of outcomes requires polynomial time.

The second result deals with cuc-acyclic CP-theories [280]. In Theorem 2, $\Delta(\alpha,\beta)$ is still used to denote the set of different variables in outcomes α and β. If $\alpha \neq \beta$, $\Theta^{'}(\alpha,\beta)$ is defined as the set of $\triangleright_{\alpha}$-undominated elements of $\Delta(\alpha,\beta)$, being $\triangleright_{\alpha}$ the transitive closure of $J_{\alpha}(\Gamma)$: $Y \in \Theta^{'}(\alpha,\beta)$ iff there exists no Z in $\Theta^{'}(\alpha,\beta)$ with $(Z,Y) \in \triangleright_{\alpha}$.

Theorem 2

Let Γ be a cuc-acyclic CP-theory, and let the binary relation $\gg_{\Gamma}$ on $\mathscr{O}$ be defined as follows, for any pair of outcomes α and β: $\alpha \gg_{\Gamma} \beta$ iff $\alpha \neq \beta$ and $\alpha(Y) \succ_{\alpha}^{Y} \beta(Y)$ for all $Y \in \Theta^{'}(\alpha,\beta)$. Then, $\gg_{\Gamma}$ is a strict partial order extending $>_{\Gamma}$, and the comparison between any pair of outcomes requires polynomial time.

Theorem 2 proposes a more general approach to generate a strict partial order on $\mathscr{O}$, which although requiring a pretty strong condition on the CP-theory, i.e., cuc-acyclicity, does not need full acyclicity. In particular, if Γ is fully acyclic, and α and β are two outcomes to compare, then $J_{\alpha}(\Gamma) \subseteq G(\Gamma)$ and so $\triangleright_{\alpha} \subseteq G^{'}$. Therefore,

$\Theta^{'}(\alpha,\beta) \supseteq \Theta(\alpha,\beta)$. This implies that if $\alpha \gg_{\Gamma} \beta$ then $\alpha \succ_{p(\Gamma)} \beta$, so that $\gg_{\Gamma}$ is a closer approximation of $>_{\Gamma}$ than $\succ_{p(\Gamma)}$.

Section 8.4.2 will ground on this more general Theorem 2 to formulate a `SPARQL` query able to rank outcomes according to user preferences encoded in a CP-theory model.

Example 4 (Books cont'd)
Relative to the CP-theory $\Gamma_{\mathrm{C-LG-SW-F}}$ with Giorgio's preferences (see Table 2.1), the use of Theorem 2 produces the following sound ranking solution:

$$\begin{aligned}
&\langle C_F LG_C SW_{No} F_{No},\\
&(C_F LG_C SW_{No} F_{Yes}, C_F LG_A SW_{No} F_{No}),\\
&C_F LG_A SW_{No} F_{Yes}, C_F LG_C SW_{Yes} F_{Yes},\\
&(C_F LG_C SW_{Yes} F_{No}, C_F LG_A SW_{Yes} F_{Yes}),\\
&C_F LG_A SW_{Yes} F_{No}, C_{UK} LG_A SW_{Yes} F_{Yes},\\
&C_{UK} LG_A SW_{Yes} F_{No}, C_{UK} LG_A SW_{No} F_{No},\\
&C_{UK} LG_A SW_{No} F_{Yes}, C_{UK} LG_C SW_{Yes} F_{Yes},\\
&C_{UK} LG_C SW_{Yes} F_{No}, C_{UK} LG_C SW_{No} F_{No},\\
&C_{UK} LG_C SW_{No} F_{Yes} \rangle,
\end{aligned}$$

where outcomes within round parentheses are not comparable.

As an example, we may consider the outcome $C_F LG_C SW_{No} F_{No}$ which is favoured in the comparisons made according to the order $\gg_{\Gamma}$ over every other outcome. In fact, the set

$$\Theta^{'}(C_F LG_C SW_{No} F_{No}, C_F LG_C SW_{No} F_{Yes})$$

coincides with the set of distinct variables in the compared outcomes, namely,

$$\Delta(C_F LG_C SW_{No} F_{No}, C_F LG_C SW_{No} F_{Yes}),$$

being both equal to $\{F\}$, and the preference φ_7 can be used to locally order the first outcome over the second one. Analogously,

$$\Theta^{'}(C_F LG_C SW_{No} F_{No}, C_F LG_A SW_{No} F_{No})$$

is equal to

$$
\begin{aligned}
& \Delta(C_F LG_C SW_{No} F_{No}, C_F LG_A SW_{No} F_{No}) \\
= \ & \{LG\},
\end{aligned}
$$

and the preference φ_5 can be exploited. The set

$$\Theta^{'}(C_F LG_C SW_{No} F_{No}, C_F LG_A SW_{No} F_{Yes})$$

coincides with

$$
\begin{aligned}
& \Delta(C_F LG_C SW_{No} F_{No}, C_F LG_A SW_{No} F_{Yes} \\
= \ & \{LG, F\},
\end{aligned}
$$

and the preferences φ_5 and φ_7 can be exploited for the variables LG and F, respectively. While

$$
\begin{aligned}
& \Delta(C_F LG_C SW_{No} F_{No}, C_F LG_C SW_{Yes} F_{Yes}) \\
= \ & \{SW, F\},
\end{aligned}
$$

it holds that

$$\Theta^{'}(C_F LG_C SW_{No} F_{No}, C_F LG_C SW_{Yes} F_{Yes})$$

is composed of the only variable SW, and the preference φ_3 can be used for this comparison. The preference φ_3 can also be used when dealing with

$$\Theta^{'}(C_F LG_C SW_{No} F_{No}, C_F LG_C SW_{Yes} F_{No}),$$

which coincides with the set

$$
\begin{aligned}
& \Delta(C_F LG_C SW_{No} F_{No}, C_F LG_C SW_{Yes} F_{No}) \\
= \ & \{SW\}.
\end{aligned}
$$

As for Giorgio, *SubsequentWork* takes priority over *LiteraryGenre* for French books, according to φ_3, and takes priority over *FilmVersion*, because of the preference φ_6 (or φ_7), it follows that the set of distinct variables

$$\Delta(C_F LG_C SW_{No} F_{No}, C_F LG_A SW_{Yes} F_{Yes})$$

has three elements $\{LG, SW, F\}$, while

$$\begin{aligned} & \Theta^{'}(C_F LG_C SW_{No} F_{No}, C_F LG_A SW_{Yes} F_{Yes}) \\ = \;& \{SW\}, \end{aligned}$$

and the preference φ_3 can be used in this case. For the last comparison involving France, the preference φ_3 is still determinant, as

$$\begin{aligned} & \Delta(C_F LG_C SW_{No} F_{No}, C_F LG_A SW_{Yes} F_{No}) \\ = \;& \{LG, SW\}, \end{aligned}$$

but $\Theta^{'}(C_F LG_C SW_{No}\ F_{No},\ C_F LG_A\ SW_{Yes} F_{No}) = \{SW\}$. When comparing the outcome $C_F LG_C SW_{No}\ F_{No}$ with any outcome o' in which C_{UK} appears, $\Theta^{'}$ contains only the undominated variable $Country$, and the preference φ_1 can be used to advantage $C_F LG_C SW_{No} F_{No}$ over o'. ■

Chapter 3

Recommender Systems

3.1 Introduction

In this section, we provide an overview of the different Recommender Systems approaches. We present the most prominent families of Recommender Systems to draw the necessary background to face the remainder of the dissertation. As a representative of Collaborative Filtering techniques, we focus first on Neighborhood-based and Matrix Factorization models. Then, Vector Space Model is detailed as a representative of Content-based models. The section is closed by an overview of the main Recommender Systems evaluation methodologies with particular emphasis on evaluation protocols and metrics.

3.1.1 The recommendation problem

Recommender Systems are software tools and techniques devoted to providing suggestions to users. The idea is to provide the user with a personalized shortlist of items or services she could appreciate. The most common recommendation strategies rely on the content (Content-Based Recommender Systems) or collaborative information (Collaborative-Filtering Recommender Systems). Beyond these families, there is still a wide range of other approaches. Graph-based, context-aware, Semantic-aware, and more recently Deep Neural Network Recommender Systems are only some examples. Taxonomies of Recommender Systems are often defined considering the source data or the recommendation technique. First, we focus on Collaborative-Filtering algorithms (CFs). On the one hand, CFs have shown the highest accuracy performance. On the other hand, we do not need to introduce any other data source. Indeed, most of them are entirely based on the transactions (or ratings) matrix. The rating matrix, usually denoted by a capital R, is a matrix that contains all the transactions that happened on the recommendation platform. These transactions could be ratings, purchases, clicks, or other feedback provided explicitly or implicitly by the users. The core idea of CFs is to exploit this information to recommend items similar users like or items that usually are consumed along with the already experienced ones. Content-Based (CBF) Recommender Systems usually do not use collaborative information. CBFs take advantage of a certain representation of items to compute similarities among them or building the user profile or both. Recommender systems, in their simplest form, can, therefore, be defined as tools that provide ranked lists of items of a specific category, based on explicit or implicit information that relies on user tastes and by exploiting additional information on items, other users, context and/or the past history of the user to whom we want to provide a recommendation list.

Up to the Netflix prize [34], the research community defined the recommendation problem as a rating prediction task. In a few words, the problem was to predict a likeness value for an unexperienced item based on the user's history. However, in real recommender systems applications, only a small subset of relevant items is provided to users [124]. Indeed, several studies pointed out that rating prediction optimization was not able to produce the optimal top-N recommendation lists [172]. The recommendation problem was hence re-defined as a top-N recommendation task [69], in which the optimization goal shifted to items ranking. In general, a good formalization of the recommendation problem comes from [237]: Let

$U = \{u_1, u_2, ..., u_m\}$ be the set of all users, and let $I = \{i_1, i_2, ..., i_n\}$ be the set of all possible items that can be recommended. Each user u_i has a list of items I_u, which the user has expressed her opinions about (explicitly as a rating score or implicitly derived from purchase records). Hence, we define an utility function of item i for user u, as $f : U \times I \rightarrow R$, where R is a completely ordered set. The recommendation problem corresponds to finding the item $i^{max,u} \in I$ that maximizes the utility function f for each user $u \in U$. Formally:

$$\forall u \in U,\ i^{max,u} = \text{arg max}_{i \in I}\ f(u,i)$$

The fundamental problem is that the utility value is not known for each item $i \in I$. We only know the utility values for the items the user has already rated in his history. Thus, the recommender system goal is predicting the utility function value on unknown data. Once we learn the utility function parameters, it is possible to realize a system that deems items' scores that are not yet been rated by the user.
It is straightforward that RSs need different kinds of data to produce their recommendations. In particular, three fundamental elements underpin any RS: **users**, **items**, and **ratings**. These components are usually represented using a rating matrix $\mathbf{R} : Users \times Items$ where each row corresponds to a user, whereas each column denotes an item. It is straightforward to infer that each matrix element is a rating given by a specific user to a specific item. In most of the cases, these matrices are very sparse because each user experienced only a small portion of the platform's items. Figure 3.1 shows a typical User-Item matrix in which question marks represent the lack of a rating. Let us draw, in detail, the three kinds of data:

- **Items** are an abstract representation of the objects or services proposed by the platform. In [177], the authors proposed an interesting taxonomy to categorize them. Atomic items can be considered as low complexity items. A few examples are news, webpages, books, songs, and movies. On the other side, some items are more complex since they can be a composition of other entities. Moreover, the complexity of an item can also vary because of the recommendation technique. In a common CF-RS, items are usually defined using only a numeric ID. Instead, in a CBF-RS, items can be extensively described by exploiting a language devoted to Knowledge Representation.

	Item 1	Item 2	Item 3	...	Item n
User 1	2	3	?	...	5
User 2	?	4	3	...	?
User 3	3	2	?	...	3
...	...	...	...	...	...
User m	1	?	5	...	4

Figure 3.1: Example of a rating matrix

- **Users** are the entities that receive recommendations and consume them. Each recommendation technique has to face at least two important problems regarding users. The first is how to deduce the behavior of users. On some platforms, where users provide an explicit rating to the items, the answer could seem obvious. In other scenarios, the behavior of users is deduced by clicks, listenings, plays, or other implicit feedback. The second problem is how to represent the user. This representation is dependant on the recommendation technique. For instance, a Neighborhood-based RS represents a user exploiting her rating vector. More, a matrix factorization algorithm uses a vector of latent factors. Finally, a CBF-RS usually represents a user with a weighted features vector.

- **Ratings**, or in general transactions, represent the relation between users and items. As mentioned, this relation can be established exploiting explicit feedback or implicit signals. A user can express explicit feedback in multiple ways. One example is the binary like/dislike rating. In this case, the user expresses her likeness of the item in a very strong and polarized way. Another example is the exploitation of a scale of values (e.g., from 1 to 5) in which the relation between the user and the item is weighted by a likeness degree. Finally, in the videos or music domain, explicit feedback is the number of playings of the video or the song. Explicit feedback comes with the advantage of avoiding misinterpretations of user behavior [205]. Indeed, with implicit feedback, this could occur at any moments, since the transformation of feedback in likeness is on the system designer. However, explicit feedback comes with an important drawback. Users are less prone to spend their time rating an item. Thus, only a small fraction of consumed items are effectively rated [134]. On the other hand, implicit feedback usually can catch only positive feedback (purchases, plays, listenings), without any information about what user dislikes. For this reason, this kind of ratings is usually also called 'unary' ratings. To sum up, explicit feedback can be considered more robust, even though they usually show inconsistencies due to users' behavior. Finally, in some cases, explicit and implicit information are combined for more accurate results [148].

3.1.2 From Rating Prediction to Ranking

As mentioned above, the recommendation task can be defined typically in two different ways. In rating prediction task, the aim is estimating the items rating for a user. Instead, Learning to Rank is the problem of recommending to users a shortlist [69] of Top-N recommendations [253]. These differences lead to different optimization goals. The former goal is modeled as a maximization of the prediction accuracy, whereas the latter is focused on generating the best list to provide to the user. Top-N recommendation task is also called **Item recommendation task** [218] since the optimization focus shifts from ratings to items. Learning to Rank [57] algorithms can be further categorized in Point-wise [151], Pair-wise [218, 169] and List-wise [245, 244]. It is important to underline that ratings are crucial in both tasks. However, one main difference is that, in a Rating Prediction task, the value associated with the user-item pair is an estimation of the rating. On the other side, in a Top-N task, the same value is used only to sort the recommendation list. We could say that the value is an estimation of a value associated with the position of the item, with no direct relation with the corresponding rating.

3.2 Collaborative-Filtering Recommender Systems

We have stressed that Recommender Systems techniques are usually classified into two main categories based on the kind of data they use to compute recommendations: Collaborative-Filtering and Content-Based. Beyond these classical models, a more exhaustive dissertation on recommendation techniques is provided in [223]. Since many of the proposed approaches are hybrid, Burke's classification [51] of hybrid Recommender Systems closes the section.

The basic idea behind a Collaborative-Filtering Recommender System is to take advantage of the transactions similarities among users or items [239]. Since the popularity of items among similar users is a very strong signal, this class of recommendation algorithms has become one of the most influential and successful. In general, these algorithms only need the user-item matrix with the corresponding feedback information. The underlying assumption is that similar users show similar preferences. Since the algorithms take advantage of users transactions, the performance of all the Collaborative-Filtering techniques is heavily affected by the number of users. The rationale behind this class of algorithms is to replicate a classic human behavior: asking similar people for reliable suggestions. Collaborative-Filtering algorithms can be further categorized in Memory-Based, and Model-Based Recommender Systems [46]. Memory-Based Recommender Systems directly exploit the user-item transactions to compute recommendations. The most extensively used approach for Memory-Based Recommender Systems is the K Nearest Neighbors (k-NN). In particular, they compute a similarity matrix to identify the more similar entities (users or items). The choice of the entities for which the similarity is computed defines the 'schema' of the k-NN algorithm. Instead, in Model-Based Recommender Systems, the algorithm trains a model exploiting the stored transactions. Usually, this model contains a latent representation of items and users. This representation is then used to produce recommendation lists.

3.2.1 Neighborhood-Based Models

User-Based k - Nearest Neighbors

This technique has been proposed by GroupLens Usenet article recommender [222] and resumed later by Ringo music recommender [243] and BellCore video recommender [127]. It computes the similarity between two customers and uses the preferences of the most similar users to estimate ratings to be assigned to items. In order to determine which user are "similar", similarity functions need to be defined. They will be detailed in Section 3.2.1. There are several ways for computing rating predictions in user-based approaches. A popular one is shown in the equation below where $r_{u,i} \in \{1,...,5\}$ is the known (five stars) rating of user u for the item i and $\hat{r}_{u,i}$ is the system prediction of the rating of u for i:

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{u_j \in N_i(u)} (r_{u_j,i} - \bar{r}_u) \cdot w_{u,u_j}}{\sum_{u_j \in N_i(u)} |w_{u,u_j}|}$$

Here $\bar{r}_u$ denotes the average rating of user u, $N_i(u)$ is the set of neighbors who rated item i and w_{u,u_j} is the similarity of the user u and u_j. The user to user similarity can be computed using different approaches. A popular one is the *Pearson correlation* [222]:

$$w_{u,u_j} = \frac{\sum_i (r_{u,i} - \bar{r}_u) \cdot (r_{u_j,i} - \bar{r}_{u_j})}{\sqrt{\sum_{i=1} (r_{u,i} - \bar{r}_u)^2} \cdot \sqrt{\sum_{i=1} (r_{u_j,i} - \bar{r}_{u_j})^2}}$$

Item-Based k - Nearest Neighbors

As the number of users increases, user-based kNN suffers from scalability problem. To overcome to this drawback, item to item kNN was introduced by Sarwar et al. in [237] and [141]. Thus, when the number of users exceeds the number of items, as is it most often the case, item-based recommendation approaches require much less memory and time to compute the similarity weights than user-based ones, making them more scalable. They are based on the similarity computation between items instead of users but the metrics used to compute similarity are the same used in user-user case. If two items shares same ratings, positive and negative ones, by the same users than they can be considered similar and hence can be assumed that a users rates in a similar way similar items. It should be noticed that, when a new rating is inserted in the platform, it impacts only peripherally on item-item similarity while in a user-based approach, users that may not have been similar since now, could become a few moments later since ratings are constantly entered in the platform. Moreover, the increasing number of users will not affect real time operations since the system will use the item-item similarity matrix to make estimations.

Similarity measures

Various similarities have been suggested, namely the Euclidean Distance, the Cosine Distance and the Jaccard similarity, among others. Which one works best in Collaborative Filtering recommender systems is openly argued, having some authors claim that the Pearson Correlation is the best suited algorithm for User based approaches [46, 123], and others identifying the Cosine Distance on item based approaches as the best one [133]. The similarity weights have two main tasks in kNN approaches: they allow to select the best and trusted neighbors and they are used to weight the candidate items from these neighbors in a different way. Among all the similarity functions, two of them are particularly important for this dissertation: *Cosine similarity* and *Jaccard similarity*. The first is computed as:

$$w_{u,u_j} = \frac{\sum_{i \in I_{u,u_j}} r_{u,i} r_{u_j,i}}{\sqrt{\sum_{i \in I_u} r_{u,i}^2 \sum_{i \in I_{u_j}} r_{u_j,i}^2}}$$

where w_{u,u_j} represents the similarity of users u and u_j, across all items commonly rated. The second is:

$$w_{u,u_j} = \frac{\sum_i^n b_{i,u} b_{i,u_j}}{\sum_i^n b_{i,u} - \sum_i^n b_{i,u_j} + \sum_i^n b_{i,u} b_{i,u_j}}$$

where $b_{i,u}$ is a binary value that is 1 when item i has been rated by the user u and n is the number of items in the overall platform.
The two previous formulas refer to a user-based case, but it is straightforward to derive the item based counterpart.

Advantages

Advantages of these methods should be highlighted:

- **Simplicity:** Neighborhood-based methods are intuitive and relatively simple to implement. In their simplest form, only one parameter(the number of neighbors) has to be chosen.
- **Efficiency:** Compared to other recommender systems that need to be retrained entirely, memory-based kNN may give an immediate feedback to users. In fact, in these systems not only similarity can be precomputed but also the K-neighbors for each item. In such a way the search operation and the prediction can be estimated in a rapid manner.

Drawbacks

However the neighborhood approaches have several flaws too:

- **Limited coverage** because of sparsity: The similarities between two users are computed by comparing their ratings for the same items. This can lead to an information loss. For example if user's u neighbors never rated item i this item will be never recommended to u despite her preferences.
- **Overfitting:** When no real neighbors can be computed, the recommender system will find them anyway leading to wrong prediction.
- For some set of neighbors the notion of rating notation in the referring space could be a mistaken assumption and the presence of neighbors could lead to erroneous estimations.
- No relationships between neighbors are considered. For example, movie's sequels will be considered and computed independently giving more weight to that kind of movie.
- **Cold Start:** at the heart of the kNN algorithm the user rating lays. Without ratings, no recommendation can take place for any user of the system. This problem named cold start, describes the inability of a recommender system to start offering recommendations when new users or items newly added to the system may have no ratings at all.

Neighborhood models summary

Building a kNN recommender system have three main steps: **normalization** of data, neighbors **selection** and interpolation weight **definition**. For each of these components, several different alternatives are available. It has to be noticed that the best approach may differ from one recommendation problem to another.
The **normalization** of data is important for each CF approach. Each user has personal interpretation of the rating scale. While one rater might tend to give high marks to films he likes, another rater might keep the highest grades for exceptional movies. There are two widely used systems to compensate for variations in the rating approach by different users, one is called mean-centering and the other is called Z-score.
The neighbors **definition** has a key role for recommendations as it emphasizes the relations between users and items. For this task, it is of paramount importance the similarity function choice. Last component is the interpolation weights **definition** because it affects how to consider item or user neighbors.

3.2.2 Matrix Factorization - SVD & SVD++

From memory-based to model-based Recommenders

Model Based algorithms received a significant push in the research community after the million dollar prize competition opened by Netflix in October 2006. During the three years that took the competition to be won, more and more contenders adopted Model Based approaches as part of their strategy. For the first time in history, the research community gained access to 100 milions movie ratings imparting a boost in new recommender systems approaches partly thanks to the nature of the competition. In fact, 1 milion dollars were offered to anyone who could improve over Netflix existent system in term of accuracy. During this competition model-based CF have proven to be more accurate than memory-based one [148]. These models create an offline model and apply it online to compute recommendations, which usually leads to greater accuracy and stability in recommendations. Although a great variety of models belong to this group, such as graph-based approaches and probabilistic recommendation approaches, the most popular are the matrix factorization ones, also called latent factor models. These models analyze the user-item matrix in order to find latent factors, which can be described as latent features that characterize the user-item relationships. Since latent factors are computed in a non-supervised way, their interpretation is not trivial, and in some cases it is not possible at all (this is the reason why they are called latent). The computation of latent factors is performed through matrix factorization techniques: the user-item matrix is decomposed into two smaller matrices, the product of which is an approximation of the original matrix. Items are then recommended to users if they are close in the latent factor space and ratings are computed through element-wise products between user and item vectors. The first approaches used the Singular Value Decomposition (SVD) to decompose the user-item matrix. On the one hand, the obvious drawback of the Model Based method is that the model, not only takes substantially longer to be computed but needs to be computed anew if the matrix of data changes, which happens every time a new user enters a rating. Generally, small changes are left unprocessed, but when they become substantial, the model needs to be retrained. On the other hand these techniques tries to overcome to two major problems: sparsity of matrix and cold start. The first one is the sparsity of matrix caused by the insufficient number of the transactions and feedback data. The user-item matrix is usually almost entirely empty because only a small part of items, w.r.t. the entire item dataset, are rated by each user. The second one concerns the personalized recommendations for users with no or few past history(new users). Providing recommendations to these users becomes a difficult problem because their learning and predictive ability is limited. Multiple research have been conducted in this direction using hybrid models. These models use auxiliary information(side information) to overcome the cold start problem. The dual problem also occurs when a new item is introduced. Since no user have previously rated this item, it will not be recommended to anyone.

Baseline predictors

Since typical CF data exhibit large user and item biases, these need to be encapsulated within the *Baseline predictors* that depends only on user and item without involving interactions between them. For instance, some users have systematic tendencies to give very high/low ratings w.r.t. to others and some items to receive very high/low ratings. It is really important to model these biases meticulously because it helps to isolate exactly interactions between users and items to compare with other users' ones. We denote the baseline of a user u on an item i as b_{ui}. The simplest baseline that can be computed is the overall average rating: $b_{ui} = \mu$. The previous formula can be improved by computing the average of the specific user or towards that specific item. By also considering the observed deviation of user u and of item i from the average, the baseline could be improved. In general a baseline predictor for an unknown rating r_{ui} is computed as in the equation below:

$$b_{ui} = \mu + b_u + b_i$$

where b_u and b_i are the aforementioned deviations for user u and i respectively. These can be estimated by solving a least squares problem but an easier way to compute them is by decoupling the calculation of the b_i's from the calculation of the b_u's; it should be noticed that this can lead to less accurate results. They can be computed as follows:

$$b_u = \frac{1}{|I_u|} \sum_{i \in I_u} (r_{u,i} - \mu)$$

$$b_i = \frac{1}{|U_i|} \sum_{u \in U_i} (r_{u,i} - b_u - \mu)$$

A regularized version of the baseline can be obtained by using the regularization parameters βu e βi:

$$b_u = \frac{1}{|I_u| + \beta u} \sum_{i \in I_u} (r_{u,i} - \mu)$$

$$b_i = \frac{1}{|U_i| + \beta i} \sum_{u \in U_i} (r_{u,i} - b_u - \mu)$$

Considering the temporal dynamics in baseline predictors could be a successful approach. So just to finalize, these baseline predictors may catch the real user bias, useful to obtain a new user-item matrix containing only the effective interactions between users and items.

Matrix Factorization Models

The most successful latent factor models are based on matrix factorization(MF)[150]. In its basic form MF characterizes both items and users by vectors of factors inferred from items rating patterns. These methods have recently become popular by combining good scalability with predictive accuracy. They offer much flexibility for modeling various real-life applications. The classic SVD model, widely used in Information Retrieval[74], is not well suited for CF domain. In fact, applying SVD to explicit ratings shows critical issues related to the high portion of missing values. One possible way to overcome to this problem is to fill the matrix with some estimations but this can lead to an inaccurate system[145, 235]. For such a reason, systems that directly model only the observed ratings have been deployed and regularization has been introduced to prevent overfitting.

SVD - Singular Value Decomposition

Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space. The latent space tries to explain ratings by characterizing both products and users on factors automatically inferred from user feedback[223]. These factors could represent both existing properties such has the genre of a movie or as is customary, completely unknown and that may not be interpreted by a human being. As a consequence, each item i is represented with a vector $q_i \in R^f$ while each user with a vector $p_u \in R^f$. Therefore, each item i will have a measure that characterizes it in each f dimensions. Same for user u in the same f dimensional space. In particular, for the user, the latent feature will represent her interest towards that property while for the item it will measure how much that property belongs to the item. The dot product $q_i^T p_u$ will capture the user's estimated interest in the item. The Figure 3.4

represents the basic form of matrix factorization. Finally, the estimated rating can be computed by combining it with the baseline predictor previously defined:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$

Our cost function to minimize will be the regularized squared error in the equation below:

$$\min_{b_*, q_*, p_*} = \sum_{(u,i) \in K} (\mu + b_i + b_u + q_i^T p_u)^2 + \lambda_4 (b_i^2 + b_u^2 + \|q_i\|^2 + \|p_u\|^2)$$

The regularization constant λ_4 is usually computed by cross validation while for the learning parameters b_u, b_i, p_u, q_i stochastic gradient descent or alternating least squares is performed. It is important to notice that alternating least square techniques fixes p_u to minimize q_i or viceversa. In this way the resulting cost function is convex and can be optimally solved[29, 28].

SVD++

Prediction accuracy could be improved through implicit feedback, useful to exploit additional information about user interest especially when few explicit ratings are provided. Moreover, in the event that no implicit feedback is available, it can be captured by considering items that user rated without considering the rating value. This lead to the development of several techniques [148, 207, 232]. Here we focus on the SVD++ method that modifies the previous equation of the estimated rating as follows:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \left(p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j \right)$$

where y_j is a generic factor vector related to the item i based on implicit feedback. Moreover the user component has been modified in $p_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j$. The sum of factor vectors y_j is normalized by $|R(u)|^{-\frac{1}{2}}$ in order to bring the variance across the range of observed values of $|R(u)|$. The cost function can now be minimized as mentioned in the SVD case. Several kind of implicit feedback may be considered and they will be automatically weighted during the minimization step.

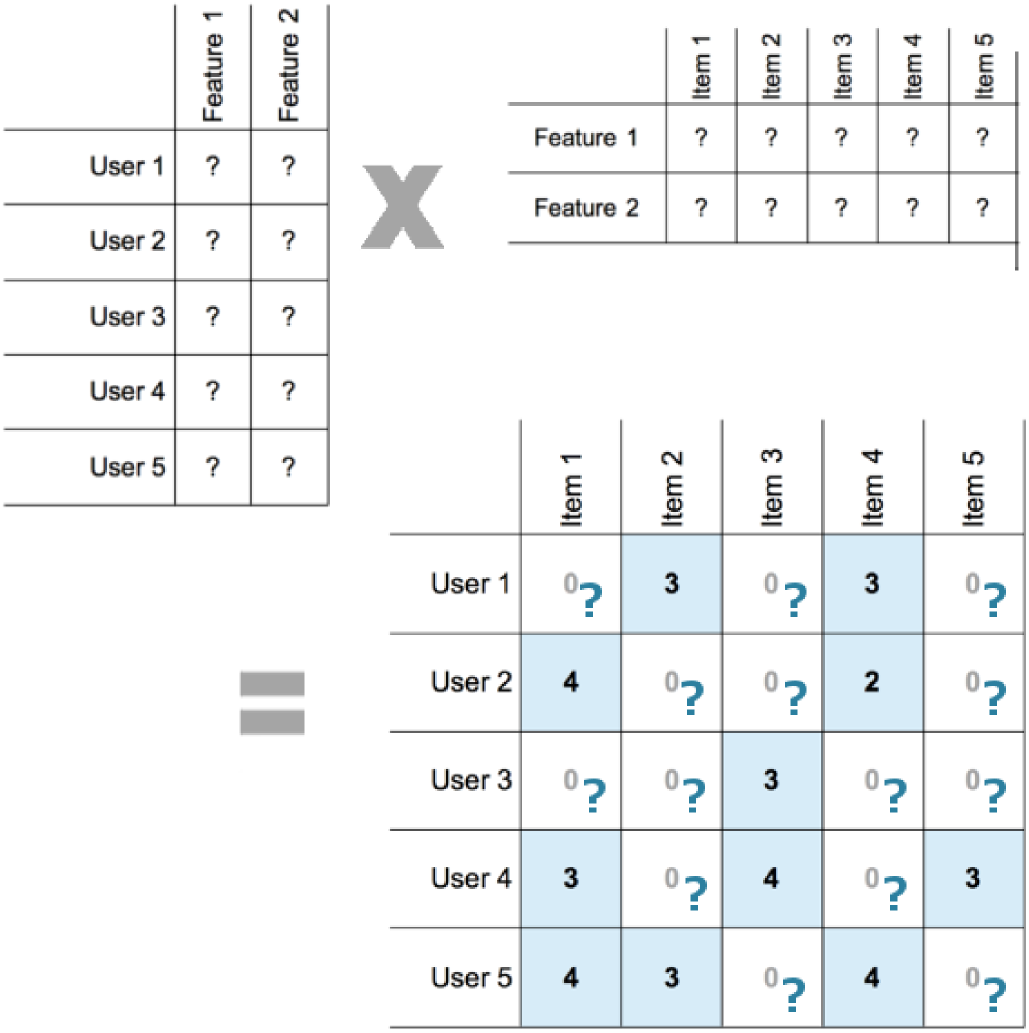


Figure 3.2: Basic representation of matrix factorization

3.2.3 Advances in Matrix factorization

Matrix Factorization (MF) models are among the most popular approaches to collaborative filtering, and have been actively investigated since they were introduced in the context of the Netflix prize competition [27]. As opposed to classic user- and item-based collaborative filtering heuristics [121, 163], MF methods train a statistical model from the available data using machine learning techniques. Specifically, they perform a *dimensionality reduction* of the highly sparse rating matrix into a subspace of latent factors, which aim to capture implicit properties of users and items. In order for MF to be effective, the dimension k of the latent subspace is assumed to be much smaller than the number of users and items, $k \ll \min(|U|, |I|)$, essentially acting as a *bottleneck* that compresses the sparse input while retaining enough information to explain the observed user-item interactions.

Matrix factorization models for rating prediction

Recommendation models based on MF have their roots on the *Latent Semantic Analysis* (LSA) technique [74], widely used in Natural Language Processing and Information Retrieval. LSA attempts to automatically infer concepts implicit in text documents by approximating the term-document matrix with a truncated Singular Value Decomposition (SVD) of lower rank. The first MF approaches for recommendation borrowed the same idea, and applied it to the user-item matrix in the rating prediction task [235]. In contrast to LSA, the SVD is not well defined for sparse matrices as those commonly found in recommender systems, and hence the above approaches relied on imputation techniques to fill the missing matrix entries before applying SVD.

Rather than filling the rating matrix, which may introduce inaccurate information, subsequent approaches aimed to only factorize observed ratings instead of the whole matrix. One of the first and most popular methods in this line is the model proposed by [101], in which each user u is assigned a vector $\vec{p}_u \in \mathbb{R}^k$ of latent features automatically inferred from the data, and similarly each item i is assigned a vector $\vec{q}_i \in \mathbb{R}^k$ in the same subspace. Intuitively, latent features aim to capture properties implicit in the data —such as the amount of *comedy* or *action* in the case of movies—, but does not need to be interpretable at all, as this is not enforced in the model [149]. Ratings are then estimated as the dot product of latent feature vectors:

$$\hat{r}(u,i) = \langle \vec{p}_u, \vec{q}_i \rangle \tag{3.1}$$

Equivalently, the rating matrix $\mathbf{R}$ is factorized as $\mathbf{R} \approx \mathbf{P}\mathbf{Q}^\top$, where $\mathbf{P}$ is a $|U| \times k$ matrix with the user vectors $\vec{p}_u$ as rows, and respectively $\mathbf{Q}$ is $|I| \times k$ contains the $\vec{q}_i$ as rows. The values of these matrices are automatically estimated from the data, by minimizing the Mean Squared Error of the ratings predicted against the ratings observed in a training set. That is, $\mathbf{P}$ and $\mathbf{Q}$ are chosen to minimize to following loss function:

$$\mathscr{L}(\mathbf{P},\mathbf{Q}) = \sum_{(u,i)\in\mathscr{R}} (r_{ui} - \langle \vec{p}_u, \vec{q}_i \rangle)^2 + \lambda \left(\|\vec{p}_u\|^2 + \|\vec{q}_i\|^2 \right) \tag{3.2}$$

where $\mathscr{R}$ is the set of observed ratings, i.e. the set of non-zero entries of the rating matrix $\mathbf{R}$, and $\lambda > 0$ is a regularization hyper-parameter used to prevent overfitting.

3.2.4 Matrix Factorization - Funk MF

In [101] this function is minimized using *Stochastic Gradient Descent*, a widely used optimization technique that iteratively updates the parameters in the opposite direction of the gradient. When applied to 3.2, this technique yields the following update rules for the parameters $\vec{p}_u$ and $\vec{q}_i$ for each rating r_{ui} in the training set:

$$\vec{p}_u \leftarrow \vec{p}_u - \eta \left(e_{ui}\vec{q}_i + \lambda \vec{p}_u \right) \tag{3.3}$$

$$\vec{q}_i \leftarrow \vec{q}_i - \eta \left(e_{ui}\vec{p}_u + \lambda \vec{q}_i \right) \tag{3.4}$$

The *learning rate* η is a hyper-parameter that controls the extent to which the model parameters are updated in each iteration, and is carefully chosen; too large values may make the algorithm fail to converge, while too small values may make its convergence very slow. e_{ui} is the prediction error, and is defined as $e_{ui} \triangleq r_{ui} - \hat{r}(u,i)$.

In addition to Stochastic Gradient Descent, other optimization techniques have been explored in the literature, such as *Alternating Least Squares* [28], which is the standard technique followed in MF models for positive-only feedback (3.2.5).

The basic SVD model by [101] is easily extensible, and has served as a building block for more complex matrix factorization models. For instance, [148] proposed the SVD++ model, which includes additional parameters to account for implicit feedback in rating predictions. Further extensions of SVD introduce temporal variables to capture the evolution of user preferences through time [149].

3.2.5 Matrix Factorization for positive-only feedback

The core ideas behind the standard Matrix Factorization model for collaborative filtering have also been applied to the item ranking task when positive-only feedback is available instead of numeric ratings. Recommendation models designed for this type of data must take into account its particular characteristics, most notably the absence of negative feedback, but also the possible uncertainty in the positive feedback, as an observed user-item interaction may not necessarily indicate a preference of the user towards the item.

In one of the most representative works in this direction, [130] proposed an adaptation of the rating-based MF model described previously to deal with positive-only feedback. As opposed to the rating-based SVD, which only considers the observed ratings, Hu et al.'s method models the full set of $|U| \cdot |I|$ interactions. Since negative feedback is not available in this scenario, the authors argue that the algorithm has also to model the missing information as an indirect source of negative user preferences. For such purpose, they introduce a parameter c_{ui} for each possible user-item pair that measures the confidence on the corresponding interaction, whether observed or not:

$$c_{ui} = 1 + \alpha k_{ui} \tag{3.5}$$

where k_{ui} is the count of implicitly collected interactions between user u and item i –such as number of clicks on a product web page on an e-commerce site, and the number of listening records of a given song in an online music provider–, and $\alpha > 0$ is a scaling parameter. When no interaction is observed, $k_{ui} = 0$ and the model assigns minimum confidence to the user-item pair, as it is unknown whether the lack of interaction is because the user really does not like the item, or just because the user does not know the item. Likewise, the more interactions are collected and the greater k_{ui}, the larger is the confidence on that observation. Moreover, focusing on the item ranking task, Hu et al.'s approach only aims to predict if the user will interact with the item, rather than the actual number of observations k_{ui}. Hence, a new set of variables is introduced so that $x_{ui} = 1$ if $k_{ui} > 0$, and $x_{ui} = 0$ otherwise.

Similarly to the SVD model for ratings, the recommendation score of item i for user u is estimated as the dot product of their corresponding latent feature vectors:

$$s(u,i) = \langle \vec{p}_u, \vec{q}_i \rangle \tag{3.6}$$

The model parameters $\vec{p}_u$ and $\vec{q}_i$ are again automatically learned by minimizing the mean squared error for the score predictions, but now accounting for the different confidence levels and the full set of possible user-item pairs:

$$\mathscr{L}(\mathbf{P},\mathbf{Q}) = \sum_u \sum_i c_{ui} \left(x_{ui} - \langle \vec{p}_u, \vec{q}_i \rangle \right)^2 + \lambda \left(\|\mathbf{P}\|^2 + \|\mathbf{Q}\|^2 \right) \tag{3.7}$$

3.3 Optimization techniques

3.3.1 ALS - Alternating Least Squares

The loss function can be minimized with different numerical optimization techniques such as Stochastic Gradient Descent, but in [130] the authors propose an *Alternating Least Squares* (ALS) procedure that efficiently handles the greater cost of accounting for the missing values. Clearly, the loss function in 3.7 involves many more terms than that of 3.2, as the number of observed entries in the user-item matrix is usually very small due to the data sparsity.

The key observation behind ALS is that when one set of parameters is fixed, the optimization problem in 3.7 is convex and analytically solvable using ordinary least-squares estimation. In particular, fixing the $\vec{q}_i$ parameters and setting the gradient with respect to $\vec{p}_u$ to zero yields the solution

$$\vec{p}_u = \left(\mathbf{Q}^\top \mathbf{C}^u \mathbf{Q} + \lambda \mathbf{I} \right)^{-1} \mathbf{Q}^\top \mathbf{C}^u \vec{x}_u \,. \tag{3.8}$$

where $\mathbf{I}$ is the $k \times k$ identity matrix, $\mathbf{C}^u$ is a $|I| \times |I|$ diagonal matrix with the c_{ui} values, and $\vec{x}_u$ is a column vector of length $|I|$ with the x_{ui} values. The same procedure

procedure ALS-TRAIN

Initialize $\mathbf{P}, \mathbf{Q}$ at random **repeat**

P step

Fix $\mathbf{Q}$ and optimize all $\vec{p}_u$ in parallel using 3.8

Q step

Fix $\mathbf{P}$ and optimize all $\vec{q}_i$ in parallel using 3.9

until *convergence*;

Algorithm 1: Alternating Least Squares training algorithm.

can be applied by fixing the user factors, and optimizing the item factors, leading to the solution

$$\vec{q}_i = \left(\mathbf{P}^\top \mathbf{C}^i \mathbf{P} + \lambda \mathbf{I}\right)^{-1} \mathbf{P}^\top \mathbf{C}^i \vec{x}_i \,. \tag{3.9}$$

Similarly, $\mathbf{C}^i$ is a $|U| \times |U|$ diagonal matrix with the c_{ui} confidence values, and $\vec{x}_i$ is a column vector of length $|U|$ containing the binary values of x_{ui}.

As pointed out by the authors, the products $\mathbf{Q}^\top \mathbf{C}^u \mathbf{Q}$ and $\mathbf{P}^\top \mathbf{C}^i \mathbf{P}$ require time $\mathcal{O}(k^2|U|)$ and $\mathcal{O}(k^2|I|)$ for each user and item, respectively, and represent a computational bottleneck during the training phase. However, these terms can be computed more efficiently noting that $\mathbf{Q}^\top \mathbf{C}^u \mathbf{Q} = \mathbf{Q}^\top \mathbf{Q} + \mathbf{Q}^\top (\mathbf{C}^u - \mathbf{I})\mathbf{Q}$, where $\mathbf{Q}^\top \mathbf{Q}$ is independent of the user and thus can be precomputed, and $\mathbf{C}^u - \mathbf{I}$ only has non-zero entries in the diagonal for the $|I(u)|$ items with $k_{ui} > 0$, which is much smaller than $|I|$. Considering the computation of the matrix inverse, the total complexity of 3.8 for a single user is $\mathcal{O}(k^2|I(u)| + k^3)$. Likewise, the complexity for 3.9 is $\mathcal{O}(k^2|U(i)| + k^3)$.

The main advantage of ALS is that the optimal factors for each user in Equation (3.8) can be computed in parallel once the item factors are fixed (P step). Symmetrically, once the user factors are obtained and fixed, the item factors in 3.9 can be found for each item in parallel (Q step). This observation leads to the alternating nature of ALS, respectively fixing one set of parameters and optimizing the other until convergence is reached or for a given number of iterations, as illustrated in 1.

The ALS-based method by [130] became the standard baseline for matrix factorization models with positive-only feedback, and has been extended and improved in subsequent works since it was first proposed. One notable paper by [213] presents a new training procedure to boost the time complexity of the P step of each user to $\mathcal{O}(k^2 + k|I(u)|)$, and analogously the Q step. In order to achieve this significant improvement, the authors propose an approximate solution to the least-squares problem in each step. Rather than directly finding the k-dimensional solution as in Equations (3.8) and (3.9), which involves the costly computation of a matrix inverse, their approach iteratively solves k one-dimensional least squares problems, one for each latent dimension, much less expensive to solve. As reported in the paper, the loss of accuracy due to the approximate algorithm is small compared to the saved time for training. In subsequent work, [259] extended ALS to a ranking-based MF approach that learns to predict the relative ordering of items instead of individual point-wise scores. More recently, [204] proposed a graph-based Bayesian model that is able to capture the meaning of missing values, distinguishing between a user disliking an item or being unaware of it.

3.3.2 BPR - Bayesian Personalized Ranking Criterion

Matrix Factorization models can be easily trained to reduce the prediction error via gradient descent methods, alternating least-squares (ALS) and MCMC. However, in a *top-N* recommendation task, MF models can be trained using a learning to rank approach like Bayesian Personalized Ranking Criterion (BPR) [218]. The BPR criterion is optimized using a stochastic gradient descent algorithm on a set D_S of triples (u,i,j), with $i \in I^u$ and $j \notin I^u$, where I^u is the set of items for the user u. Each triple is selected through a random sampling from a uniform distribution. The BPR optimization criterion can thus be formulated as:

$$\begin{aligned} \texttt{BPR-OPT} &= \sum_{(u,i,j)\in D_s} ln\sigma(\hat{x}_{uij}) - \lambda_\Theta \|\Theta\|^2 \\ &= \sum_{(u,i,j)\in D_s} ln\sigma(\hat{y}(\mathbf{x_{ui}}) - \hat{y}(\mathbf{x_{uj}})) - \lambda_\Theta \|\Theta\|^2 \end{aligned} \tag{3.10}$$

where $\hat{x}_{uij}$ is the estimated rating difference, and $\hat{y}(\mathbf{x_{ui}})$ is the prediction the user-item pair $u-i$. In this formulation, $\sigma(\cdot)$ is a sigmoid function, and the update step is defined as:

$$\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda \Theta \right) \tag{3.11}$$

where α is the chosen learning rate. In an implicit feedback setting, we may assume that there is only an instance for the pair user-item. Hence, in the model we can derive $\hat{x}_{uij}$ as:

$$\begin{aligned} \hat{x}_{uij} &= \hat{y}(\mathbf{x_{ui}}) - \hat{y}(\mathbf{x_{uj}}) = w_i - w_j + \\ &+ \sum_{f \in F} v_{(u)} v_{(i)} - v_{(u)} v_{(j)} \end{aligned} \tag{3.12}$$

where w_i is the bias for the given item i, and $v_{(u)}$ is the vector of latent factors for the user u. This computation can be performed in an efficient way computing the partial derivatives (to update the factorized parameters) for the only active entities involved in the transactions, w_i, w_i, v_u, v_i, and v_j:

$$\frac{\partial}{\partial \Theta} \hat{x}_{uij} = \begin{cases} 1, & \text{if } \theta = w_i, \\ -1, & \text{if } \theta = w_j, \\ v_{(u)}, & \text{if } \theta = v_{(i)}, \\ -v_{(u)}, & \text{if } \theta = v_{(j)}, \\ \left(v_{(i)} - v_{(j)}\right), & \text{if } \theta = v_{(u)}, \\ 0, & \text{otherwise} \end{cases} \tag{3.13}$$

Using Equation (3.13) in Equation (3.11) the model parameters can be iteratively updated to maximize the `BPR-OPT` criterion. The algorithm updates sequentially each sampled triple and continues the training until it reaches the provided number of iterations.

3.4 Content-Based Recommender Systems

Content-based filtering Recommender Systems exploit item content to provide recommendations based on what the users rated in the past. The recommendation pro-

cess starts with the definition of a user profile. While items often have a technical description in terms of features, the challenging task is to extract significant features that can drive the recommendation process. In content-based methods, feedback of users is combined with the content information available in the items. In most cases, the items' attributes are simple keywords that are extracted from their description. Semantic indexing techniques are also used to represent the item and user profiles using concepts instead of keywords. The recommendation process basically consists of matching up the attributes of the user profile against the attributes of a new item. The result is a relevance estimation that represents the user's level of interest in that object. If a profile accurately reflects user preferences, the effectiveness of a content-based approach is considerably improved. The most common technique to define a user profile from the descriptions of the items she liked is to extract a list of words that describes such items. In this case, the main issue is the definition of the criteria to apply for selecting the most important words. Therefore, thanks to the availability of several open knowledge sources such as DBpedia[1] and to the increasing interest in semantic technologies, lot of research works moved from the classic keyword-based approach to a concept-based one.
The architecture of a content-based recommender system is composed of three main components: **Content Analyzer**, **Profile Learner**, and **Filtering Component**. Figure 3.3 shows the high-level architecture of a content-based recommender system.

The *Content Analyzer* is a component devoted to extracting the relevant features of an item. It exploits feature extraction techniques to build a new item representation that could be processed in the simplest way by a recommender system.

However, this item could be of any kind and this is the reason why this task could lead to a very difficult issue to solve. For example, analyzing a document or a web page may be a simple task while in the case of a movie or an image it might not be that easy. In the first case, for instance, an approach that takes into account most relevant words could be adopted by using some metrics that define the significance of a word in that document or web page. It is straightforward that the choice of

[1] http://wiki.dbpedia.org/

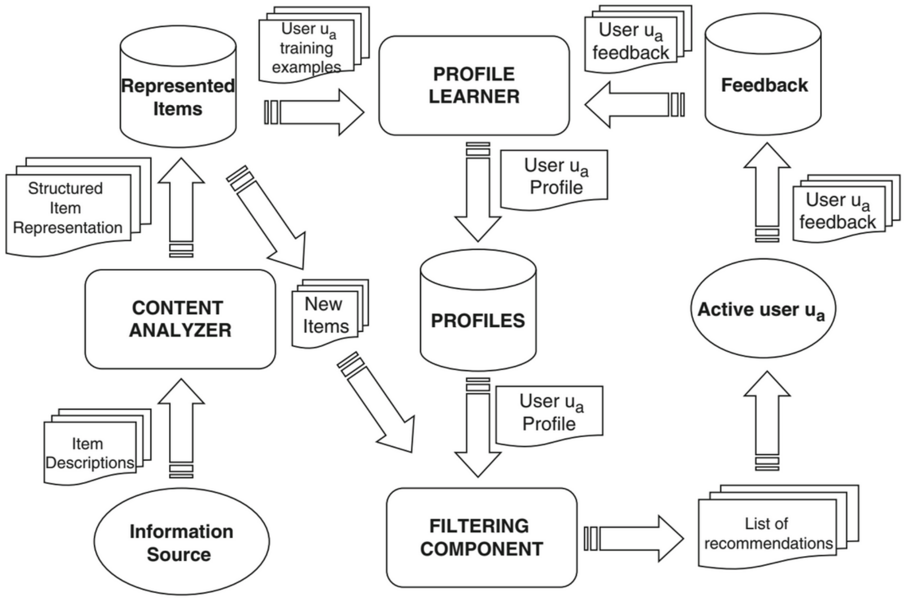


Figure 3.3: High level architecture of a content-based recommender [223]

approach affects the performance of the entire recommender system.

For movies or images, the analysis becomes more complex. Besides visual features extraction, the problem can be alleviated if some description of those items is available in a knowledge base. Then, we can exploit this structured information to feed the recommender system. However, once a description is retrieved, it is still important to state what information should be considered.

The new representation of the item becomes the input of the *Profile Learner* and *Filtering Component*
The *Profile Learner* exploits the descriptions to match with the abstract user model. Its goal consists in inferring user preferences and interests to build a user model. Usually, Machine learning techniques are employed to train the user model. Finally, the *Filtering Component* is responsible for estimating a score for a generic item exploiting the user profile representation. This is the component that produces the list of recommendations.

Conclusion

The adoption of the content-based recommendation paradigm has several advantages:

- Content-based recommenders are based only on ratings provided by the history of the target user and on their ability to learn a user profile representation that approximates user preferences as well as possible. This means that no collaborative information is needed to feed the recommender. Instead, a Collaborative-Filtering method needs ratings from other users to provide recommendations. Consequently, Content-based recommenders do not suffer the so-called Cold-start items problem.

- They can provide an understandable and immediate *explanation* of the recommended items thanks to the nature of the recommendation process. Indeed, *Explanations* can be provided, in their simplest form, by explicitly listing content features.

Content-based methods do have several shortcomings as well:

- Even though content-based methods are effective at providing recommendations for new items, they are not effective at providing recommendations for new users. The reason is the dependency of the user profile from the history of her ratings. In general, a large number of ratings implies a robust representation of the user profile and hence robust predictions.

- As mentioned before, content-based recommender systems are based on the past of the user to suggest new items that are consistent with it. This approach will tend to produce always the same kind of recommendations over time. As an example, a person that has not experienced romantic movies would never receive a recommendation for a romantic movie. This behavior is called *over-specialization* problem or lack of *Serendipity*. These terms highlight the tendency of the content-based systems to not produce recommendations with unexpected items.

- Best Content-based techniques require something more than factual information, they require domain knowledge. As an example, the description of a movie can distinguish among actors and the director, and this can lead to accuracy improvements. Typically, the elected tool to let a recommender understand structured information is providing domain ontologies. Unfortunately, domain ontologies are not always available. Consequently, feeding the recommender with only factual information can lead to losing some important information for the recommendation task.

- The vast size of the item set is a problem that content-based methods have to face. Since we need to find items that correlate the most with the user's interests, we are forced to examine all the items. Moreover, we must examine the content of every item to estimate a score, whereas collaborative filtering systems only need to examine users ratings. Therefore, since the number of items rises very quickly, a content-based suffers scalability issues. As a result, the solution is simplifying the item representation. Commonly, this leads to a performance decrease.

3.4.1 Vector Space Model

Many content-based recommender systems represent documents using a simple spatial representation. Probably the most know representation is the Vector Space Model (VSM). In a VSM, each user or item (document) is represented in an n-dimensional space, where n is the cardinality of the considered features (or keywords). In practical terms, users and items are represented through Boolean or weighted vectors. Their respective positions and the distance, or better the proximity, between them, provides a measure of how these two entities are related or similar. The choice of features may substantially differ depending on their availability and application scenario: crowd-sourced tags, categorical, ontological, or textual knowledge are just some of the most exploited ones. All in all, in a CB approach we need:

1. to get reliable items descriptions;

2. a way to measure the strength of each feature for each item;
3. to represent users;
4. to measure similarities.

Item descriptions The set of available features is usually generated processing the textual description of items. Some of the most common operations in this sense are: tokenization, stopwords removal, and stemming, or even more advanced Natural Language Processing methods [24].

Feature strength The different terms within a document usually deserve various degrees of importance for categorizing the document. What we do need is a weighting procedure, a scheme, that lets us assign the correct weight to different terms. Some of the most adopted schemes are TF-IDF and BM25. TF-IDF, in particular, can be derived from the probabilistic distribution of terms, and it reflects some common observations [233]. First of all, rare terms should not be considered less relevant than frequent terms. Second, if a term is frequent in a document it is not less relevant than occurring once. Last, the contribution of each document in the weighting should be independent of the length of the description.

TF-IDF Given a set of items $I = \{i_1, i_2, \ldots, i_N\}$ in a catalog and their associated features f_i in a Collection $\mathscr{C}$ we may build the set of all possible features as $F = \{f \mid f_i \in \mathscr{C} \text{ with } i \in I\}$. In the following, we use f to denote a feature in F irrespectively of the item. Let us denote with $freq(f,i)$ the frequency of the feature f in the document i. Consequently, $max_z freq(z,i)$ denotes the maximum frequency considering all the features of item i. Let us $I(f)$ represents the subset of I which considers only those items that contain f. Each item can be then represented as a vector of weights $\omega_i = [\omega_{(i,1)}, \ldots, \omega_{(i,f)}, \ldots, \omega_{(i,|F|)}]$ where $\omega_{(i,f)}$ is computed as the normalized TF-IDF value for f:

$$\omega_{(i,f)} = \underbrace{\frac{\frac{freq(i,f)}{max_z freq(z,i)}}{\sqrt{\sum_{k \in F} \left(\frac{freq(i,k)}{max_z freq(z,i)} \cdot \log \frac{|I|}{|I(k)|} \right)^2}}}_{TF} \cdot \underbrace{\log \frac{|I|}{|I(f)|}}_{IDF}$$

User Representation Analogously, when we have a set U of users, we may represent them using the features describing the items they enjoyed in the past. Given a user u, if we denote with I^u the set of the items enjoyed by u. Let us denote with $I^u(f)$ the set of items experienced by u which contain f. Hence, we have $\omega_u = [\omega_{(u,1)}, \ldots, \omega_{(u,f)} \ldots, \omega_{(u,|F|)}]$ with

$$\omega_{(u,f)} = \frac{\sum_{i \in I^u} \omega_{(i,f)}}{|I^u(f)|}$$

Similarity As mentioned, a similarity measure is required to measure the closeness between users and items in the features space. The proximity of two vectors can be computed in multiple ways. However, a very common strategy to evaluate similarities between the vectors i and j is evaluating the cosine vector similarity (CSV) of their corresponding vectors in $\mathbf{F}$:

$$CSV(i,j) = \frac{\omega_i \cdot \omega_j}{\| \omega_i \| \cdot \| \omega_j \|}$$

3.4.2 From Vector space Model to Knowledge-aware Recommender Systems

Regarding the descriptions of items, nowadays we can easily retrieve them from the Web. In particular, thanks to the Linked Open Data initiative a lot of semantically structured knowledge is publicly available in the form of Linked Data datasets. In [191], the authors originally proposed to encode a Linked Data knowledge graph in a vector space model to develop a CB recommender system. Let us recall that, in a Linked Data dataset, a resource is described using triples in the shape *subject-predicate-object*, where the subject is the resource itself. Accordingly to the graph

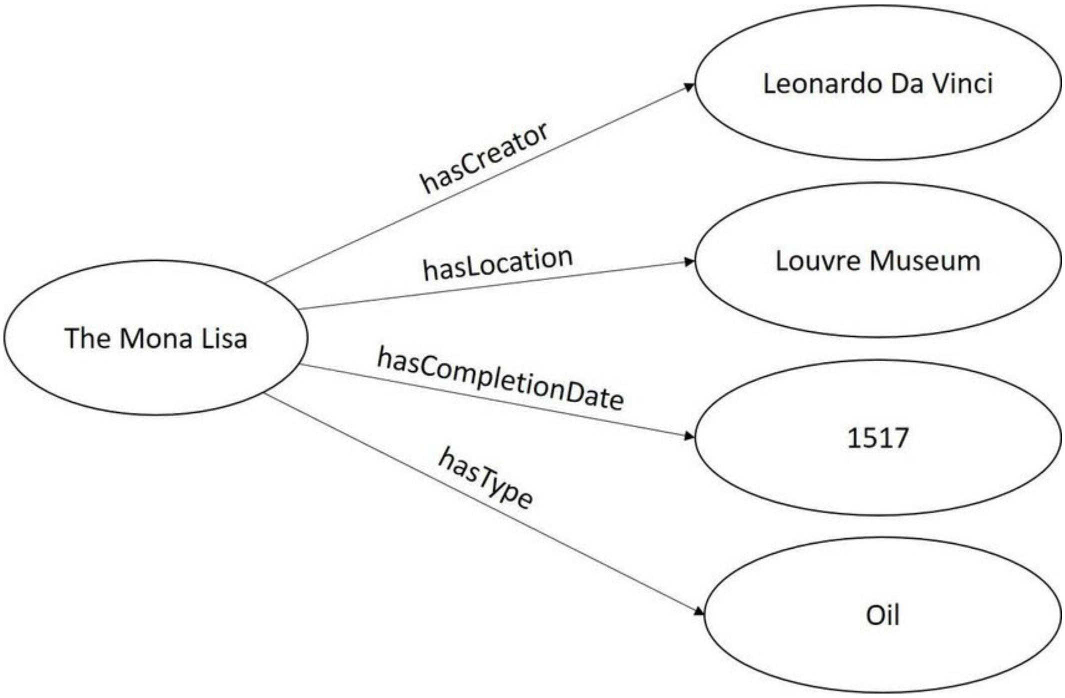


Figure 3.4: Basic example of item semantic description

nature of data, the underlying problem is building the vectors ω_i, and ω_u which represent, respectively the item and the user vectors. This difference requires to redefine the features representation. Given a set of items $I = \{i_1, i_2, \ldots, i_N\}$ in a catalog and their associated triples $\langle i, \rho, \omega \rangle$ in a knowledge graph $\mathscr{KG}$ we may build the set of all possible features as $F = \{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathscr{KG} \text{ with } i \in I\}$. In the following, when no confusion arises, we use f to denote a feature $\langle \rho, \omega \rangle$ in F. Each item can be then represented as a vector of weights $\mathbf{i} = [v_{(i,1)}, \ldots, v_{(i,\langle \rho, \omega \rangle)}, \ldots, v_{(i,|F|)}]$ where $v_{(i,\langle \rho, \omega \rangle)}$ is computed as the normalized TF-IDF value for $\langle \rho, \omega \rangle$:

$$v_{(i,\langle \rho,\omega \rangle)} = \underbrace{\frac{|\{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathscr{KG}\}|}{\sqrt{\sum\limits_{f \in F} |\{f \mid \langle i, f \rangle \in \mathscr{KG}\}|^2}}}_{TF^{\mathscr{KG}}} \cdot \underbrace{\log \frac{|I|}{|\{j \mid \langle j, \rho, \omega \rangle \in \mathscr{KG} \text{ and } j \in I\}|}}_{IDF^{\mathscr{KG}}}$$

Analogously, when we have a set U of users, we may represent them using the features describing the items they enjoyed in the past. Given a user u, if we denote with I^u the set of such items and we have $\mathbf{u} = [v_{(u,1)}, \ldots, v_{(u,f)} \ldots, v_{(u,|F|)}]$ with

$$v_{(u,f)} = \frac{\sum\limits_{i \in I^u} v_{(i,f)}}{|\{i \mid i \in I^u \text{ and } v_{(i,f)} \neq 0\}|}$$

Once we have generated the new item and user representation, the computation of the user-item similarity (and hence the generation of recommendations) can proceed as aforementioned.

3.5 Hybrid Recommenders

Hybrid Filtering is a combination of different Filtering approaches [197]. The rationale is proposing a filtering method able to go beyond the limitations of traditional approaches. Some of these limitations are the Cold-Start problem, the over-specialization of Content-Based approaches and the sparsity of the Users-Items matrix. Moreover, some hybrids recommenders aim directly to increase the accuracy of recommendations and performance of the whole system. The goal of exploiting a hybrid recommender is to compensate for the limitations of an approach using the advantages of another. There exist multiple ways to implement this kind of recommender [7]:

- Implementing the two approaches separately and then use an aggregation function to merge the results;
- Injecting some Content-Based approach characteristics into a Collaborative-Filtering approach;
- Injecting some Collaborative-Filtering approach characteristics into a Content-Based approach;
- Building a single system that exploits at the same time collaborative and content information.

Typical examples of hybrid recommenders are streaming platforms like Netflix or Amazon Prime. On one side, they provide recommendations based on collaborative information. On the other side, they also exploit movie features to propose

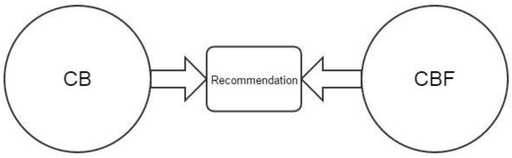


Figure 3.5: Hybrid system that implements the two approaches separately and aggregates the results

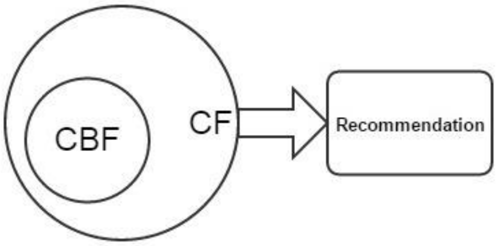


Figure 3.6: Hybrid system that injects some Content-Based approach characteristics into a Collaborative-Filtering approach

more accurate recommendations. In detail, in 2006 Netflix released an anonymized dataset regarding one million users challenging researchers to overcome their recommendation algorithm. Among the best approaches, most of them were simple hybrids that had taken advantage of ensembles of different methods.

It is straightforward that the choice of the integration scheme affects the resulting architecture. Figure 3.5 shows that the two recommenders are completely distinct and the integration takes place only during the generation of the recommendations. Figures 3.6 and 3.8 show that we can make one of the recommenders completely dependant on the other, which usually becomes the new knowledge source. Figure 3.7 depicts a single system that is fed by two different information. In this classification scheme, there are no constraints about the exploitation of that information.

Another interesting classification for hybrid models has been proposed by Burke [50]:

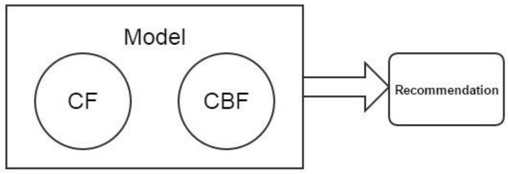


Figure 3.7: Hybrid system that exploits at the same time collaborative and content information

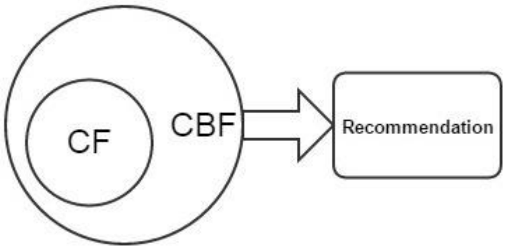


Figure 3.8: Hybrid system that injects some Collaborative-Filtering approach characteristics into a Content-Based approach

- **Weighted:** The ratings returned by the different approaches are then aggregated through a weighting function;
- **Switching:** The system evaluates the context and based on that it selects the best algorithm to propose recommendations;
- **Mixed:** The different algorithms generate different recommendation lists. These lists are then merge using an arbitrary mechanism like the mean or a combination of parts of the lists;
- **Feature combination:** The weights of the features obtained after the training are used as the new knowledge source. Another recommender receives this input and proposes the recommendations;
- **Cascade:** The recommendation process is considered as a sequential process in which the output of a recommender feeds another;

- **Feature augmentation:** The output of a recommender is used to enrich items and users' descriptions of another recommender;

- **Meta-level:** The overall system exploits one recommendation algorithm to generate the model and this model is used as input for another recommender.

3.6 Recommender Systems Evaluation

In the last decades, we observed a flourishing of Recommender Systems algorithms. Most of them are very specialized approaches, designed to perfectly fit a specific scenario. The same recommender fed with different data would probably perform poorly. This happens because identifying what makes Recommender Systems behave differently is a hard task. For instance, it could be due to operating in different settings. However, even the degree of sparsity of the Users-Items matrix can affect the performance of the system. Moreover, even though two systems operate in the same scenario, with the same data, identifying the causes of the different behaviors would be hard. Indeed, there is no total convergence on which is the most important dimension of evaluation, neither of which is the best metric to measure [124].

Evaluating a recommender can be hard for many reasons. As an example, given the same set of data, an algorithm can show oscillating performance based on the evaluation target. Moreover, different evaluation settings can affect the evaluation result. Finally, even if we consider the recommendation accuracy, measuring the rating prediction error is a completely different task from evaluating the ranking of recommended items.

In [172], the authors have pointed out that accuracy is not the only aspect to consider during evaluation. Others consider the online evaluation as the only effective evaluation. However, online evaluation with real users means bearing the costs of the platform and involving users. Even not considering the necessary evaluation time, it is clear that this kind of evaluation is not affordable for every research group. For these reasons, we focus on offline evaluation. This section is devoted to providing a brief overview of the evaluation protocol, metrics, and methods [124].

3.6.1 Protocols

How to assess the quality of a recommendation list, even though we are limiting our analysis to offline evaluation, is not an easy task. It could be argued that the main aspect to consider is the accuracy of the recommendations. However, the definition of accuracy is not unique. For instance, if we change the recommendation task from rating prediction to items' recommendation, also the notion of accuracy should be re-formulated. Before going into the different definitions, we need to give some Recommenders Evaluation background. Bellogin [153] has shown that there are many aspects in common between the Recommender Systems and Information Retrieval evaluation. This is reasonable since the research field was born as a branch of Information Retrieval. Moreover, if we focus on a Top-N recommendation task, the similarities are more evident. Consequently, most of the evaluation techniques and metrics we use were originally designed for Information Retrieval tasks. However, between the two evaluation paradigms, there are important differences. First of all, in Information Retrieval domain experts can objectively evaluate the relevance of a document. This is not possible in a recommendation scenario.

The relevance of a recommended item can be established only on a per-user basis. This has dramatic effects on the evaluation. In theory, we should evaluate a Recommender System only online, proposing recommendations and waiting for the feedback. However, this means that a study conducted by a small/medium-sized laboratory can rely only on a few users. Moreover, the number of items that are effectively rated is usually very small. How a researcher should deal with the remaining unrated items is an open problem. Indeed, the choice of ignoring them or considering them as negative examples affects either the training phase and the evaluation procedure. Moreover, the splitting of data in a training set and a test set can be realized in multiple ways. Here too, the choice has significant effects on the evaluation results [253]. The authors underline that, even though the test set contains only a small number of items, the ranking affects the entire collection. For this reason, they propose two different evaluation protocols: **all unrated items**, and **rated test-items**.

The **all unrated items** protocol imposes to use as candidate items all the items that have never been rated by the user. A candidate item is an item that can be potentially recommended. Consequently, the items that are present in the test set and the recommendation list are considered as relevant [31][231].. The rationale is that in a real case scenario the recommender analyzes all the items in the collection. Instead, in the **rated test-items** protocol, the candidate items correspond to test items. The difference among algorithms is measured by ranking ability or prediction errors.

3.6.2 Accuracy

As mentioned, the accuracy metrics are broadly categorized in error metrics, and ranking metrics. According to the historical development of metrics, first we focus on error metrics.

Rating prediction metrics

In the past, the metrics based on prediction error have been the most widely used. Among them, the most known ones are the **Root Mean Squared Error (RMSE)** and **Mean Absolute Error (MAE)**. The goal of a **rating prediction task** is to estimate the rating for an item by a given user. Once the recommender is trained, we use its prediction function to estimate the rating value $\hat{r}_{ui}$, for the test set (TS) item. For this user-item pair (u,i), the actual value of rating is known but hidden to the recommender.

$$\text{MAE} = \frac{1}{|TS|} \sum_{(u,i) \in TS} |\hat{r_{ui}} - r_{ui}| \tag{3.14}$$

$$\text{RMSE} = \sqrt{\frac{1}{|TS|} \sum_{(u,i) \in TS} (\hat{r_{ui}} - r_{ui})^2} \tag{3.15}$$

Ranking Metrics

Since the research community has moved toward the **Top-N recommendation task**, the previous metrics have fallen out of use. The reason is that a low rating error does not imply a correct ranking for Top-N recommendations. Indeed, these metrics evaluate errors of all relevant items without considering any sort order. Consequently, the error of the last relevant item is equally important to the error on the most relevant one. To sum up, rating prediction metrics may estimate the same error for top-N items and bottom-N items, without taking into account that an error in top-N items should be more relevant compared to an error for lower ranked items.

The new evaluation need has pushed researchers to propose new metrics. Among them, we there are **Precision**, **Recall**, and **Normalized Discounted Cumulative Gain**. Since these metrics check the presence of items in the first **N** recommendations, the metric value is coupled with a number which denotes the list length. Common values are: 1, 5, 10, 25, 50, 100. Precision and Recall require binary values. In practical terms, they assign 1 to relevant items and 0 otherwise.

Given a user u, Precision ($P_u@n$) is computed as the ratio of the number of relevant items in the recommendation list over N. Recall ($R_u@N$) is computed as the ratio of the number of relevant items in the recommendation list over the number of relevant items in the Test Set.

Precision is defined as the proportion of retrieved items that are relevant to the user.

$$P_u@N = \frac{|L_u(N) \cap TS_u^+|}{n} \tag{3.16}$$

where $L_u(N)$ is the recommendation list up to the N-th element and TS_u^+ is the set of relevant test items for u. Precision measures the system's ability to reject any non-relevant documents in the retrieved set. Recall is defined as the proportion of relevant items that are retrieved.

$$Recall@N = \frac{|L_u(N) \cap TS_u^+|}{TS_u^+}$$

Recall measures the system's ability to find all the relevant documents.

where TS_u^+ is the set of relevant items in the Test Set for user u and $L_u(N)$ is the recommendation list truncated at N. Precision is heavily affected by the evaluation protocol. On the other side, Recall is affected by the threshold chosen to identify a test item as relevant. The items with an actual rating lower than the threshold are considered non-relevant. For this reason, this threshold is usually called the relevance threshold. When a relevance threshold is used to compute a metric, it is adopted the *threshold-based relevant items* condition [52] Precision and recall can be combined with each other in the F1 measure computed as the harmonic mean between precision and recall.

$$F1@N = 2 \cdot \frac{Precision@N \cdot Recall@N}{Precision@N + Recall@N}$$

In information retrieval, Discounted cumulative gain (DCG) is a metric of ranking quality that measures the usefulness of a document based on its position in the result list. Recommended results may vary in length depending on the user, therefore is not possibile to compare performance among different users, so the cumulative gain at each position should be normalized across users. Let us define r_{uk} as the rating assigned by the user u to the item at the position k of the recommendation list. Hence, normalized discounted cumulative gain, or `nDCG`, is computed as:

$$nDCG_u@N = \frac{1}{IDCG@N} \sum_{k=1}^{N} \frac{2^{r_{uk}} - 1}{\log_2(1+k)} \tag{3.17}$$

where k is the position of an item in the recommendation list and $IDCG@N$ indicates the score obtained by an ideal ranking of the recommendation list $L_u(N)$ that contains only relevant items. $IDCG@N$ is used as a normalization factor. However, for `all unrated items`, there is no rating for the majority of items. A common adopted solution [253] is defining a single fix value instead.

3.6.3 Diversity

Accuracy metrics are needed to measure performance of a top-N recommender. Nevertheless, recommendation quality also depends on other factors that could improve the user experience, such as Novelty and Diversity.

Diversity, as an alternative dimension of the performance of recommender systems, can be measured through common measures as *item coverage*, *catalog coverage*, *Gini index*, and *Shannon entropy*.

Some recommenders could be designed to produce recommendations only for some users or recommend only a small number of items. This problem is known as the long tail or heavy tail problem. Nevertheless, if recommendations include very popular items, the accuracy of recommendations can still be high. On the other side, the interest of users toward only very popular items could be limited. These considerations have pushed researchers to define new metrics that take into account these problems. They have proposed two metrics: **User Coverage**, and **Item** (or **Catalog**) **Coverage**.

Coverage

Catalog Coverage computes the number of items that are effectively inserted in recommendation lists. It denotes the propensity of a system to recommend always the same items. An example of measuring this quantity is **aggregate diversity**. This metric, defined as **diversity-in-top-n** in [5], measures the overall number of distinct items recommended in all recommendation lists.
ADiv@N is computed as:

$$ADiv@N = \frac{|\bigcup_u L_u(N)|}{|I|}$$

It is worth to note that in this formula it is present a normalization for the length of the catalog.

Moreover, this metric provides some information about the personalization of recommendation. A high value of Item Coverage implies that most of the users received completely different lists. Dually, a low value of Item Coverage suggests that the recommendation lists are not diversified.

Similarly, the **User Coverage** [112] denotes the overall number of different users the recommender is able to produce a recommendation for. It denotes the ability of a system to produce recommendation for all the users.

Distributional inequality

Another measure of diversity is **sales diversity** [99], which measures how the items are unequally distributed in the recommendation lists. The *Gini index* (*Gini*) and Shannon entropy (*SE*) are two different metrics used to measure the distributional inequality [59]. Distributional inequality measures how unequally different items are chosen by users when a particular recommender system is used. Considering L_I the list of items in the collection ordered in increasing order of the probability to be put in a recommendation list $p(i)$, the *Gini index* is defined as:

$$Gini = \frac{1}{n-1}\sum_{j=1}^{n}(2j-n-1)p(i_j)$$

where $i_1, ...i_n$ is the list of items sorted by increasing $p(i)$.

The index goes to 0 when all items are equally chosen. On the other side, it goes to 1 when the recommender always selects the same item. According to the evaluation target, it is possible to adopt another formulation that considers the number of times an item is recommended. Usually, the Gini index is represented through a reversed scale, obtained as 1 - Index. This choice makes the index more readable because, with this formulation, higher is better. In this case, low values correspond to a scenario in which items are not equally chosen.

Shannon entropy (SE), on the other hand, takes the same probability and computes a different value:

$$SE = -\sum_{i=1}^{n} p(i)\log p(i)$$

Entropy goes to 0 when a single item is recommended. When a recommender recommends n items, the entropy value reaches $\log(n)$.

Intra-List Diversity

Another interesting characteristic is Intra-List diversity. In general, we could say that a recommendation list is diversified if the items it contains are different. Usually, a recommender does not care about the similarity or dissimilarity of recommended items. However, let us suppose to design a recommender for a travel agency. If the different alternatives are similar, the recommender is not suggesting anything interesting to users. To improve user experience [251], we should provide a varied recommendation list. In this scenario, to measure diversity, we have to measure a degree of diversity between items in the same list [269].

Since diversity is an opposite signal to similarity, we can choose a similarity metric for this task. In literature, several strategies have been proposed: means, summations, minimum and maximum distances. An example is `Intra-List Diversity` (`ILD`) [304].

Expected Intra-List Diversity is a diversity metric which measure how much is diversificate the recommendation.

$$ILD_u@N = \frac{1}{2} \sum_{x_i \in L_u(n)} \sum_{x_j \in L_u(n)} 1 - sim(x_i, x_j) \tag{3.18}$$

$$ILD@N = \frac{1}{|U|} \sum_{u \in \mathbb{U}} ILD_u@N \tag{3.19}$$

Finally, the diversification can also be computed considering the different intents of the user. In order to evaluate the diversification power in this case we could measure ERR-IA[61].

$$ERR-IA = \sum_{r=1}^{n} \frac{1}{r} \sum_{t} P(t|q) \prod_{i=1}^{r-1} (1 - R_i^t) R_r^t$$

where r is the position of an item i, t is the topic, $P(t|q)$ is the conditional probability of the topic given the query (user profiles in recommendation scenario), R_i is the probability of the relevance of the item and R_r is the probability of the relevance of the list of items from 1 to r. With this metric, the contribution of each item in the recommendation list is based on the relevance of documents ranked above it. The discount function then also depends on the relevance of previously ranked documents.

3.6.4 Novelty

The **novelty** of recommendations [269] could be defined as the presence in the recommendation list of completely unknown items for the user [147]. There is not a unified definition of novelty. Indeed recommender systems use different definitions of novelty. However, in general, it is considered as a measure of how many new items are recommended. A common approach is measuring the number of recommended items that come from the long tail. It is a signal of the ability of a system of recommending items that never the user could have discovered. Hence, measuring novelty is crucial for a successful recommender.

A metric that measures the ability of the system of proposing long-tail items is `Entropy-Based Novelty` (`EBN`) [30]. Let us define again the recommendation list as $L_u(N)$. `Entropy-Based Novelty` can be computed as:

$$EBN_u@N = -\sum_{i \in L_u} p_i \cdot \log_2 p_i$$

in cui:

$$p_i = \frac{|\{u \in U \mid i \text{ è rilevante per l'utente } u\}|}{|U|}$$

With this *EBN* formulation, when $EBN_u@N$ decreases the novelty increases.

A metric that measures the ability of the system of proposing long-tail items is `Entropy-Based Novelty` (`EBN`) [30]. Let us define again the recommendation list as $L_u(N)$. `Entropy-Based Novelty` can be computed as: Another novelty metric is `Expected Popularity Complement` which corresponds to the expected number of relevant items that come from the long-tail. In this case, the binary relevance formulation of *EPC* [59] can be computed as:

$$EPC = C \sum_{i_k \in \mathbb{R}} disc(k) p(rel|i_k, u)(1 - p(seen|i_k)) \tag{3.20}$$

where $disc(k)$ is a discount function, $p(rel|i_k, u)$ is the relevancy of the item in the recommendation list and $(1 - p(seen|i_k)$ reflects a factor of item novelty. C is a normalizing constant, which stabilize the metric against unwanted biases. Two approaches are used in information retrieval to define $\frac{1}{C}$. The former defines $\frac{1}{C}$ as the maximum metric value obtainable by an ideal recommendation ranking as in nDCG or a-nDCG. The latter defines it as the expected browsing depth.

EFD, on the other hand, is a measure of the expected inverse collection frequency of relevant and seen items:

$$EFD_u@N = -\frac{1}{|Rec_u^N|} \sum_{i \in Rec_u^N} \log_2 p(i|seen)$$

The aforementioned metrics provide a measure of the ability of a system to recommend relevant long-tail items.

Part II

Feeding RSs with explicit knowledge

Chapter 4

Introduction

This chapter is focused on the exploitation of a formal representation of knowledge to feed Recommender Systems. We have already provided a broad overview of Semantic Web technologies and Recommender Systems techniques. Hidden in the overview, a careful reader may have read some hints on the approaches we are proposing. External knowledge can help propose more accurate recommendations. However, a fine representation of knowledge can highlight other similarities. This usually leads to improvements in terms of other dimensions like diversity and novelty. Moreover, the collaborative filtering algorithms usually deal with very sparse matrices. This lack of data can be compensated by external knowledge.

In the last decade, collaborative filtering approaches have shown their effectiveness in computing accurate recommendations starting from the user-item matrix. Unfortunately, due to their inner nature, collaborative algorithms show their limits when they deal with sparse matrices and, in these cases, encoding user preferences only through past ratings may lead to unsatisfactory recommendations. Hybrid approaches have been proposed to cope with this issue by exploiting side information about the items within the catalog. In the first line of research, we propose to inject knowledge **from semantic graphs to matrices**. In practical terms, we propose to exploit past user ratings, and Linked Open Data to evaluate the relevance of every single feature within each user profile thus moving from a user-item to a user-feature matrix. Here, each value is a pair representing both the popularity of the feature in the user profile and its estimated rating. We then propose two computationally efficient content-based approaches and two hybrids, that make use of matrix factorization techniques to compute recommendations. The evaluation has been performed on three datasets referring to different domains (movies, music, and books) and experimental results show that the proposed methods outperform state of the art approaches in terms of accuracy, novelty, and diversity of results.

Providing relevant personalized recommendations for new users is one of the major challenges in recommender systems. This problem, known as the user *cold start* has been approached from different perspectives. In particular, cross-domain recommendation methods exploit data from source domains to address the lack of user preferences in a target domain. Most of the cross-domain approaches proposed so far follow the paradigm of collaborative filtering and avoid analyzing the contents of the items, which are usually highly heterogeneous in the cross-domain setting. Content-based filtering, however, has been successfully applied in domains where item content and metadata play a key role. Such domains are not limited to scenarios where items do have text contents (e.g., books, news articles, scientific papers, and web pages), and where text mining and information retrieval techniques are often used. Potential application domains include those where items have associated metadata, e.g., genres, directors and actors for movies, and music styles, composers and themes for songs. With the advent of the Semantic Web and its reference implementation Linked Data, a plethora of structured, interlinked metadata is available on the Web. These metadata represent a potential source of information to be exploited by content-based and hybrid filtering approaches. Motivated by the use of Linked Data for recommendation purposes, in the second line of research we present and evaluate a number of **matrix factorization models for cross-domain collaborative filtering that leverage metadata** as a bridge between items liked by users in different domains. We show that in case the underlying knowledge graph connects items from different domains and then in situations that benefit from cross-domain information, our models can provide better recommendations to new users while keeping a good trade-off between recommendation accuracy and diversity.

Model-based approaches to recommendation can recommend items with a very high level of accuracy. Unfortunately, even when the model embeds content-based information, if we move to a latent space we miss references to the actual semantics of recommended items. Consequently, this makes non-trivial the interpretation of a recommendation process. In the third line of research, **we show how to initialize latent factors in Factorization Machines by using semantic features coming from a knowledge graph in order to train an interpretable model**. With our model, semantic features are injected into the learning process to retain the original informativeness of the items available in the dataset. The accuracy and effectiveness of the trained model have been tested using two well-known recommender systems datasets. By relying on the information encoded in the original knowledge graph, we have also evaluated the semantic accuracy and robustness for the knowledge-aware interpretability of the final model.

Preference representation and reasoning play a central role in supporting users with complex and multi-factorial decision processes. In fact, user tastes can be used to filter information and data in a personalized way, thus maximizing their expected utility. Over the years, many frameworks and languages have been proposed to deal with user preferences. Among them, one of the most prominent formalism to represent and reason with (qualitative) conditional preferences (CPs) are *conditional preference theories* (*CP-theories*). In the fourth line of research, **we show how to combine *CP-theories* with Semantic Web technologies in order to encode in a standard SPARQL 1.1 query the semantics of a set of CP statements representing user preferences** by means of RDF triples that refer to a "preference" OWL ontology. In particular, here we focus on context-uniform conditional (cuc) acyclic CP-theories [280]. The framework that we propose allows a standard SPARQL client to query Linked Data datasets, and to order the results of such queries relative to a set of user preferences.

In *Recommender Systems* and, more broadly, in *Information Retrieval* scenarios, the notion of relevance for the attributes of an item (or a document) plays a crucial role. As an example, all the items belonging to a recommended list are supposed to be of interest to the user because there is a similarity between the relevance of their attributes and those belonging to the items already enjoyed by the user in the past. Relevance measures such as *TF-IDF* or *BM*25 are a representation of the informativeness of attributes in item description and are based exclusively on content-based information. In this investigation, we propose to enhance pure content-based relevance values for item attributes by exploiting collaborative information. **The idea is that of representing a vector of attributes whose weights encode both content-based and collaborative knowledge in a principled way.** To show the effectiveness of our proposal, we have tested it on three different datasets with respect to *state-of-the-art* algorithms in *Recommender Systems*.

Chapter 5

From semantic graphs to matrices

5.1 Introduction

Recent years have seen the flourishing of many and diverse recommendation techniques based on collaborative information encoded in the user-rating matrix. Factorization techniques have proven their effectiveness in improving the performance of recommendation engines and are implemented in many industrial and commercial systems [137, 27]. More recently, deep learning arrived as a new player in the field to develop powerful collaborative and hybrid approaches [67]. The core idea of collaborative filtering is to exploit the user-user connections through items and ratings to estimate user tastes and then predicting a list of items the user may be interested in. State-of-the-art algorithms can capture complex non-linear or latent factors-based relationships between users and items thus resulting more effective in all those scenarios where several users partially overlap their ratings or, in other words, the user-rating matrix is less sparse. To overcome the limits of pure collaborative approaches, hybrid ones [50] have been proposed that also encode side information, typically content-based, about the items. Indeed, whenever available, descriptions of the items can be used as a valuable source of information to augment the knowledge injected in and exploited by the system to compute a recommendation list of items. In this direction, an interesting class of recommender systems is the so called semantics-aware [73] where the information describing items goes beyond text and keywords and is represented by categorical/ontological data. Semantics-aware (SA) approaches have been widely adopted to integrate domain knowledge in a recommender system. SA approaches make use of ontologies or encyclopedic sources to encode and exploit such knowledge and in the last years many approaches have been proposed [174, 40, 154]. In fact, the domain-specific knowledge eases the process of interpreting documents and extracting relevant information from them. More recently, thanks to the Linking Open Data initiative, many structured data have become freely available to represent the content of items in different knowledge domains and they have been used to feed recommendation engines [191].

As a general remark, we can say that most of the recommendation algorithms available in the literature focus on computing the relevance of a set of items with reference to the user profile. Recommendation algorithms are designed around the computation of a relevance score for an item by evaluating its similarity with reference to other items. Features composing the description of an item, whatever the source, are not considered per se in the recommendation process but are usually exploited to evaluate the similarity between items or users. We believe that more attention might be paid to modeling the recommendation problem with a focus on recommending features rather than items. Expanding an item in its features give us a new set of explicit connections between items to be exploited with collaborative filtering algorithms. Finally, recommending items via feature recommendation may lead to an easier generation of explanation for the recommended list of items.

Unfortunately, moving from items to features is not that straight as in a forest of many features, most of them may result not relevant to a user. Moreover, once we design an algorithm able to compute a recommendation list of features, we have to go back to the items space, as the ultimate goal of a recommender systems is to suggest items to a user.

In this research line we present `FF` (for `Feature Factorization`), a *top-N* recommendation algorithm relying on user's feature preferences and collaborative filtering information in the features space. The main goal of `FF` is to compute an ordered list of features preferred by the user and, starting from such list, to reassemble the relevance values of each returned feature to produce a *top-N* list of items to recommend. All the side information adopted by `FF` is retrieved from `DBpedia`, the cornerstone dataset of the Linked Data cloud. For each item in the user profile we retrieve its features by querying `DBpedia` thus having them as a set of entities. This avoids all problems related to synonymy and polysemy which usually occur when dealing with keyword-based features. By combining the popularity of a feature in the user profile and the ratings assigned to the items it is part of, for each user we compute a pair containing the relevance of the feature and its inferred rating. The resulting matrix in the user-feature space can be then manipulated via factorization techniques to compute, for each user, a ranked list of features which is in turn post-processed to produce the final list of recommendations.

Experimental evaluations of `FF` on two datasets related to the domains of books and music show its effectiveness in terms of accuracy, diversity and novelty of results in very sparse settings.

Please note that, although our approach shares some points with multi-criteria recommender systems [6], `Feature Factorization` differs from them because of the following main reasons:

- In `FF`, we do not assume any explicit rating to a specific feature of an item but we rather try to infer it starting from the global rating of the item;
- We target a top-N recommendation task and not a rating prediction one;

First, we formalize an initial version of `FF`, purely based on feature factorization. It is straightforward this version deserves its own experimental evaluation to assess if it is a feasible research direction. Then, starting from the initial version of `FF` we have worked on the following research questions:

RQ1 Are all the retrieved features needed to get results comparable with state-of-the-art algorithms?

RQ2 How much does FF performance depend on the number of items we are able to find a mapping for?

RQ3 Is it possible to reduce the computational effort of FF and keep at least the same results in terms of accuracy?

RQ4 What is the influence of the quality for Linked Data information on the recommendation results?

RQ5 How does FF perform with reference to diversity and novelty of recommendations?

The remainder of the chapter is structured as follows. In the next section we report some related work on LOD-based and feature-based approaches to recommendation. We continue in Section 5.3 by introducing and describing FF and its extensions. Experimental evaluations are presented in Section 5.4 while in Sections 5.4.1 and 5.4.2 we present and discuss the corresponding results. Conclusions and future work close the chapter.

5.2 Related Work

Several works have tried to build recommender systems by exploiting Linked Open Data (LOD) as side information for representing users or items, in addition to the user preferences usually collected through their ratings. Such approaches usually rely on DBpedia, a dataset which acts as a hub for most of the knowledge in the so-called LOD cloud. In the following, we review the recent literature on LOD-based recommender systems and, besides, since we propose an approach that leverages the relevance of single features in the user profile, we present related work in feature-based recommender systems.

LOD-based RS. Recommender Systems can exploit the knowledge coming from the LOD cloud for different tasks and in several ways. A detailed review of the literature, up to 2015, on Recommender Systems leveraging Linked Open Data is presented in [73]. Properties gathered from DBpedia may be used, e.g., to produce cross-domain recommendations [96], to build a multirelational graph for a graph-based recommender [194], or to generate effective natural-language recommendation explanations [182]. In [95] the authors propose three matrix factorization models for cross-domain recommendation. `LOD` are exploited to compute inter-domain item similarities, addressing the lack of data in the cold-start scenario. In [273] a web tool is provided, to encode the semantics of *Conditional preferences theories* [280]. Users preferences are expressed as RDF triples using a specific OWL ontology. These preferences are then used to produce meaningful recommendation lists via SPARQL queries. *ExpLOD* [182] is a novel tool able to generate natural language explanations exploiting information encoded in `Linked Open Data` encyclopedic datasets. In [179, 180] an extensive evaluation is performed, to establish whether `LOD` features are beneficial or not in using a graph-based recommender system, specifically a *PageRank with Priors* [116] algorithm. *Linked Data Semantic Distance (LDSD)* measure has been proposed by [211] to compute a new distance metric to feed `LOD`-enabled recommender systems. In [192] authors summarize the main adopted techniques to feed a recommender system with `LOD`, explaining all the crucial steps to extract this kind of knowledge and to exploit it. In [181] popularity, collaborative, and content-based information is exploited with and without Linked Data to evaluate the impact on the usage of `LOD` features to produce recommendations.

When working with a Linked Data dataset, e.g., `DBpedia`, its properties may be used in very different ways:

1. to define semantic similarity measures for providing more accurate recommendations [206, 183, 173, 211, 193];

2. to deal with classical problems of recommender systems, as the *limited content analysis* or *cold-start*, e.g., by introducing new relevant features to improve item representations [42, 240], or to cope with the increasing data sparsity [179];

3. to improve the overall accuracy of a recommender system [198, 178], or to provide a good balance between different recommendation objectives, such

as accuracy and diversity [142, 179, 196].

Feature-based RS. The most widely adopted recommendation algorithms rely on the assumption that there are sufficient historical data for measuring similarity between items or users. Unfortunately, this assumption does not hold in several domains, where new items often lack ratings or comments, and where products that are less often purchased have fewer records of ratings. Several works attempt to analyze the user purchasing behavior based on item features and, consequently, user preferences towards specific features of items are then analyzed and exploited to provide potential accurate recommendations. In [278], products are represented using vectors of features, and a customer profile module computes the level of interest of the customer in product features as the ratio of features among the products purchased, and the product quantity purchased by that customer. Euclidean distance is then used to calculate the similarity between the customer and product profiles to recommend the *top-N* products. Similarly, in [114] a feature-based recommender system is presented for domains without enough historical data to effectively measure user or item similarities. The authors build the system based on the idea that *users who bought items with specific features also buy items with the same or similar features*. A similar approach is proposed in [186], in which effective strategies to incorporate item features for *top-N* recommender systems are developed. In graph-based recommender systems, an interesting work was proposed in [117], where recommendations are produced inferring user preferences, evaluating item-preferences and attribute-preferences. The paper points out the importance of the feature evaluation and a method is proposed which exploits explicit feature ratings named attributes. Another approach called Feature Preferences Matrix Factorization (FPMF) has been proposed in [185]. FPMF incorporates user feature preferences in a matrix factorization to predict user likes. Preferences on features are interpreted as user's attributes, which are taken into account when computing predictions on items by introducing additional latent factor vectors corresponding to attributes. It is worth noticing that none of the previous mentioned approaches exploits features coming from the `Linked Open Data` cloud.

Yet, approaches on feature-based RS need suitable datasets containing the explicit opinions of users on features of items, i.e., the ground truth for the preferences of users on items' features, in order to evaluate user models in terms of matching between estimated and true preferences on features. In [266], a Synthetic Data Generator is presented able to produce user-item and user/item-attribute datasets, while in [184], the authors used crowdsourcing to collect a dataset in the movie domain containing the explicit preferences of users on both items and their attributes. In [298] authors face the problem of splitting sparse data in time dimension. Instead of using pre-assumed distributions, they exploit item features extracted from textual reviews to realize a feature-level user model. In another interesting work, User-specific Feature-based item-Similarity Models (UFSM) [88] are proposed, in which multiple global item similarity functions are computed, together with user-specific weights for each function. The personalized weighted function is then used to produce recommendation lists. In [287] authors propose ReMF, a matrix factorization algorithm able to deal with hierarchically structured features. They also propose a new regularization method, named recursive regularization. Collaborative Knowledge Base Embedding framework (CKE) is proposed in [292]. The framework is designed to collaboratively learn latent representation starting from visual, textual, and structured knowledge. Knowledge base embedding is performed using three source-specific embedding techniques: TransR, SDAE, and SCAE respectively for structured, textual, and visual sources. The three representation are then combined in a single pair-wise ranking optimization function, to learn users and items collaborative representations. In [125] parallel Recurrent Neural Network (p-RNN) architecture is used to model sessions based on features extracted from pictures and text descriptions. Another interesting work is [267], in which 3-dimensional convolutional neural networks are used to produce recommendations. Authors chose this architecture to cope with item factual and categorical features within a single session. In details, this architecture is exploited to catch spatiotemporal patterns. In Boosted Factorization Machines (BoostFM) [289] a re-weighting scheme is proposed to inject boosting technique into a Factorization-Machine-based recommender system. Authors also propose two methods for pair-wise and list-wise learning to rank optimization. In [139] explicit items and features pair-wise preferences are mapped to item comparisons in order to provide more accurate recommendations. DeepLIFT [247] technique is proposed to highlight the most important features in a neural network model. A different perspective is proposed by [72], in which the collaborative information is extracted and encoded in the form of new features to feed a content-based recommender system. Discrete Factorization Machines (DFMs) [164] are a recent proposal in which the high computational cost of feature-based and factorization machines algorithms is dramatically reduced by exploiting binarization of real-valued parameters. In order to avoid the quantization loss, the authors propose a different updating rule. The implicit short-term interests of the user are modeled through attentive neural networks in [228] with the aim of maximizing the time spent by users on the platform. Instead of polyphonic timbres of the song or topics, authors make use of embeddings of tags for the item occurred before the song to be predicted. In [224] a new graph embedding technique, RDF2Vec, is introduced, that make use of Weisfeiler-Lehman Subtree RDF Graph Kernels and graph walks to generate latent representations. More recently a Multi-Task Feature Learning approach (MKR) for Knowledge Graph Enhanced Recommendation [274] has been proposed that learn high-order interactions between items and entities in the knowledge graph. This approach takes advantage of knowledge graph embeddings and associates each item in a catalog to one or more entities in the knowledge graph. Other researchers build their approaches by leveraging user-defined features, such as tags, in order to produce better recommendations. We will not survey this line of research since, although tags seem to convey preferences over features, tagging an item does not necessarily mean that the user liked the attribute denoted by that tag. Finally, in our opinion feature-based Rs research line may have an overlap with multi-criteria RSs [6], that studies innovative approaches in collaborative recommendation by attempting to capture and model user preferences in a more comprehensive and nuanced manner by engaging multi-criteria ratings, in order to represent users' subjective preferences for various components of individual items.

5.3 Approach

5.3.1 Motivation

This line of research aims at investigating the role of feature rating and relevance in the item rating process. The main intuition behind `FF` is that items can be considered as a collection of features. Hence, when users rate an item, they are actually expressing their preference over the whole collection. In the evaluation of a movie, the user implicitly evaluates the director, the actors, the producer, the country in which the movie is set. Each feature has its own rating and a relevance degree, hence a Recommender System should consider these factors. The item rating action can be then interpreted as an attempt to choose an overall rate for the entire set of features. Our assumption is that if we want to discover the contribution of each single feature in the evaluation, first of all, we need to unpack each item in its composing features. Then, by combining the overall popularity of each feature in the user profile (feature relevance) and the rating assigned to items containing that feature we may estimate the implicit rating the user is giving to that specific feature.

In our model the user profile is not just a set of $\langle item, rating \rangle$ pairs but it contains information about the relevance (popularity) of each feature composing the rated items and its estimated rating. It is then represented as a set of triples $\langle feature, relevance, rating \rangle$. In the following, we see principled methods to estimate both the user-feature rating and the user-feature relevance, and then we move to the recommendation problem using the features that compose the user profile.

5.3.2 Data Model

For a better understanding of the data we use to reshape the user profile in terms of knowledge-aware features, we first introduce the multidimensional graph we have used to build them. As we can see from Fig. 5.1 the user profile is built by considering information coming from both the `Users-Items` matrix and from `DBpedia` as external knowledge source modeled as `Linked Data`. Now we see how the graph-based nature of this latter is exploited to identify features used to represent items. Linked Data are represented as RDF labeled oriented graphs and their data model is based on the notion of triple $\langle subject, predicate, object\rangle$ where *predicate* represents the relation connecting the two entities *subject* and *object*. With reference to Figure 5.1, we have that an item i in the catalog of a recommendation engine may represent the subject of a triple $\langle i, p, e\rangle \in$ DBpedia. As an example, in the movie domain we have triples like $\langle \texttt{Matrix}, \texttt{starring}, \texttt{Keanu\_Reeves}\rangle$, $\langle \texttt{Matrix}, \texttt{director}, \texttt{The\_Wachowskis}\rangle$. It is noteworthy that the same pair *predicate*, *object* may occur for different items as for $\langle \texttt{Point\_Break}, \texttt{starring}, \texttt{Keanu\_Reeves}\rangle$ and, at the same time, the same entity may appear as object in diverse triples as in the case of $\langle \texttt{V\_for\_Vendetta}, \texttt{producer}, \texttt{The\_Wachowskis}\rangle$ where the entity The_Wachowskis is connected to the subject via producer instead of director as for the previous triple. In order to catch the different knowledge encoded in the use of the same entity as object in triples with diverse predicates, in our model, we consider the chain $p \circ e$ as a feature associated to the item i which in turn represents the subject of the corresponding triple. In the model we propose, each item in the user profile is associated with a *relevance function* denoted with $\rho^{ui}(\cdot)$. Its value represents an estimation of how important is a particular item to the user u. Analogously, we have a value associated to each feature in the profile computed via the function $\rho^{uf}(\cdot)$ that for each feature represented by the chain $p \circ e$ returns a value representing the relevance of that feature in the user profile. As we have pointed out in Section 5.3.1, we assume there is a relation between the two relevance values which should be reflected in the mathematical formulation of the two functions $\rho^{ui}(\cdot)$ and $\rho^{uf}(\cdot)$. Actually, each feature is also associated with a *rating* $r^{uf}(\cdot)$ which is inferred by considering the rating of all the items containing the specific feature.

5.3.3 Problem Formulation

By considering the data associated to the user profile we can move from a rating matrix connecting user and items to a user-feature matrix where each value is represented by the pair $\langle \rho^{uf}(\cdot), r^{uf}(\cdot)\rangle$. In other words, we may consider two user-feature matrices: the one $\mathscr{P}$ containing relevance values $\rho^{uf}(\cdot)$, the other $\mathscr{R}$ the inferred

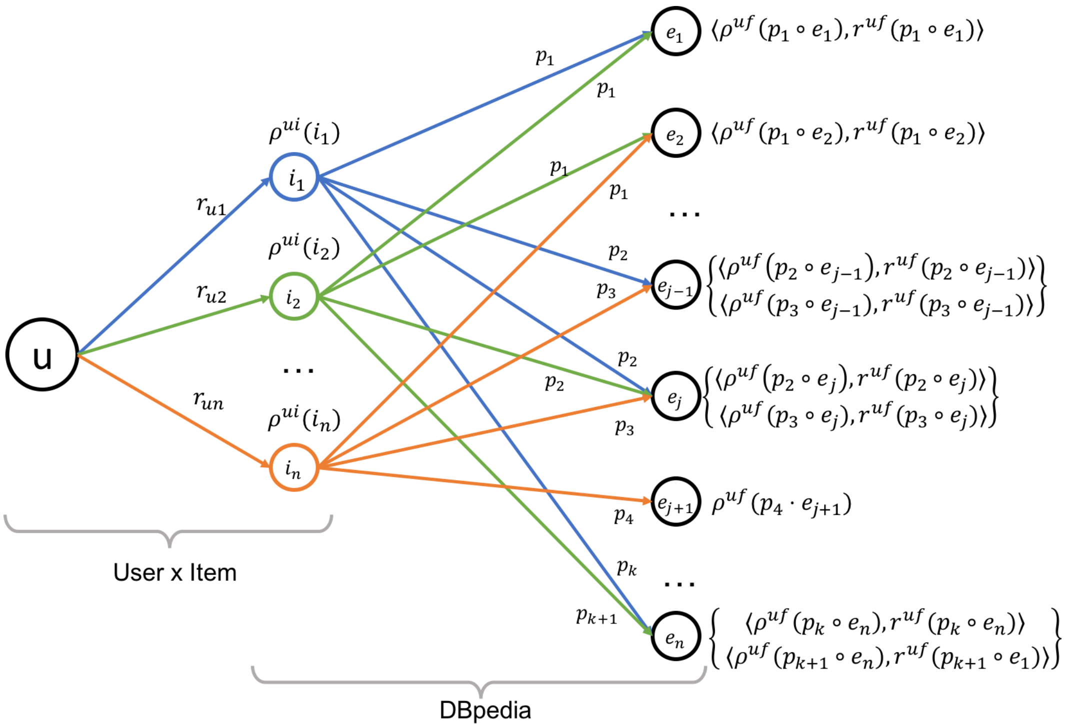


Figure 5.1: A graph-based representation of the data behind the computation of the user profile.

ratings $r^{uf}(\cdot)$. Given I representing the catalog of items and I^u containing the items experienced by the user u, for the sake of compactness, in the formulation of the problem we now introduce the following sets:

$$F^i = \{p \circ e \mid \langle i, p, e \rangle \in \texttt{DBpedia} \wedge i \in I\}$$

$$F^u = \{p \circ e \mid \langle i, p, e \rangle \in \texttt{DBpedia} \wedge i \in I^u\}$$

$$F = \bigcup_{i \in I} F^i$$

$$I^{uf}(p \circ e) = \{i \mid i \in I^u \wedge p \circ e \in F^i\}$$

In FF, the relevance of a feature $p \circ e$ is computed as its probability of belonging to I^u. More formally we have:

$$\rho^{uf}(p \circ e) = \frac{|I^{uf}(p \circ e)|}{|I^u|}$$

The idea behind this computation is quite straight: the more a feature is connected to the items in the user profile (i.e., the more a feature is popular), the higher is its relevance for the user. Once we have computed the relevance of all the features in the user profile, we can move to the computation of the relevance for the items $i \in I^u$. It can be computed as the normalized summation of the relevance for all the features it is composed by. In formulas, we have:

$$\rho^{ui}(i) = \frac{\sum_{p \circ e \in F^i} \rho^{uf}(p \circ e)}{|F^i|}$$

Please note that $\rho^{ui}(i)$ is not defined for $i \notin I^u$. As discussed before, we see that the relevance of an item is influenced by the relevance of the features it is composed by. In terms of the final user-feature matrices we want to compute, the relevance of an item is used to estimate the rating associated to the features occurring in the items already rated by the user. Given a feature $p \circ e$, the computation of $r^{uf}(p \circ e)$ exploits both the rating and the relevance of each item $i \in I^u$ containing $p \circ e$.

$$r^{uf}(p \circ e) = \frac{\sum_{i \in I^{uf}(p \circ e)} r^{ui} \cdot \rho^{ui}(i)}{\sum_{i \in I^{uf}(p \circ e)} \rho^{ui}(i)}$$

To completely move from a user-item space to a user-feature one, we combine $\mathscr{P}$ and $\mathscr{R}$ by introducing a new matrix $\mathscr{S}$ obtained by element-wise multiplication of $\mathscr{P}$ and $\mathscr{R}$. Each element of $\mathscr{S}$ is then computed as:

$$s^{uf}(p \circ e) = \rho^{uf}(p \circ e) \cdot r^{uf}(p \circ e)$$

Because of the spreading mechanism exploited by the method, we name $s^{uf}(\cdot)$ **feature spreading relevance** (`fsr`). Our intuition is that features with the higher values of `fsr` are the most representative for the users and are those influencing their ratings. To test our intuition, we have defined different similarity measures based on $s^{uf}(\cdot)$ and we then have tested them in a recommendation scenario.

5.3.4 Pure `FF`

The profiles we have built contain only the features the user has met before, but usually the number of those features is dramatically smaller than the overall number of features and this results in $\mathscr{P}$ and $\mathscr{R}$ being very sparse. To complete the information they contain, we compute, via Biased Matrix Factorization, the missing values $\hat{\rho}^{uf}(p \circ e)$ for $\mathscr{P}$ and $\hat{r}^{uf}(p \circ e)$ for $\mathscr{R}$. We run matrix factorization independently on $\mathscr{P}$ and $\mathscr{R}$. Biased Matrix Factorization is a Matrix Factorization model that minimizes RMSE using stochastic gradient descent [150]. It computes user's and item's biases to improve the estimation of the predicted value. Biased Matrix Factorization represents a state-of-the-art algorithm in rating prediction task. $\hat{\rho}^{uf}(p \circ e)$ and $\hat{r}^{uf}(p \circ e)$ represent the predicted relevance and the predicted rating for all those features not belonging to any of the items in I_u. As the resulting matrices contain both content-based and collaborative information (due to the Matrix Factorization), we refer to them as *hybrid profiles*.

With the *hybrid profile* we can estimate a ranked list for all the remaining items within the collection. In fact, the ranking of an item in the list is computed by considering the rating of the features belonging to the item and their relevance.

$$\begin{aligned}\hat{r}^{ui}(i) = & \sum_{(\langle i,p,e\rangle \in \texttt{DBpedia}) \wedge (i \in I_u)} \rho^{uf}(p \circ e) \cdot r^{uf}(p \circ e) + \\ & + \sum_{(\langle i,p,e\rangle \in \texttt{DBpedia}) \wedge (i \notin I_u)} \hat{\rho}^{uf}(p \circ e) \cdot \hat{r}^{uf}(p \circ e)\end{aligned} \tag{5.1}$$

It is important to point out that these estimations do not correspond to an actual rating. Instead, the goal is trying to preserve a correct item ranking.

Post-filtering

To improve the results of the final recommendation process, we propose a post-filtering step for reducing the number of features considered while computing the final rank. The proposed filtering springs from the following observations:

- Not all the features items are relevant in the computation of the ranking for an item. All those features whose rating results low just introduce noise in the final values we compute.

- Feature ranking and relevance values evaluated via pure content-based approaches, i.e., before the Matrix Factorization, have a different influence if compared with the collaborative ones representing latent factors computed after the Matrix Factorization.

To lower the number of features involved in the computation, and produce recommendations based only on the best ratings of the estimated features, we propose a filter that operates on directly estimated features (content-based), and estimated features coming from collaborative computation. Then, we introduce two thresholds α and β that act as filters on the feature rating values respectively in the content-based and the collaborative cases. Hence, Equation (5.1) is slightly modified:

$$\begin{aligned}\hat{r}^{ui}(i) = & \sum_{(\langle i,p,e\rangle \in \texttt{DBpedia}) \wedge (i \in I_u) \wedge r^{uf}(p \circ e) > \alpha} \rho^{uf}(p \circ e) \cdot r^{uf}(p \circ e) + \\ + & \sum_{(\langle i,p,e\rangle \in \texttt{DBpedia}) \wedge (i \notin I_u) \wedge \hat{r}^{uf}(p \circ e) > \beta} \hat{\rho}^{uf}(p \circ e) \cdot \hat{r}^{uf}(p \circ e)\end{aligned} \tag{5.2}$$

5.3.5 Jaccard-`fsr`

One of the simplest way to compute the similarity between two objects (items or users) is by Jaccard similarity. It is computed by taking into account the common elements between the two objects. In our case, if we consider a pure content-based setting, we may compute the similarity between a user and an item by looking at the common features available within the user profile u and the item description i. Actually, as the features with a higher value of `fsr` should be more relevant for u, then we may select only those having the highest value of $s^{uf}(\cdot)$ for u and check only these latter against i instead of the whole user profile. To identify these valuable features, we have first computed $\mu(u)$ representing the average value of $s^{uf}(\cdot)$ for u and we eventually select only those features with a value of `fsr` higher than $\mu(u)$. More formally, based on the following definitions:

$$\mu(u) = \frac{\sum_{p \circ e \in F^u} s^{uf}(p \circ e)}{|F^u|} \tag{5.3}$$

$$Rel^{u+} = \{p \circ e \mid p \circ e \in F^u \wedge s^{uf}(p \circ e) > \mu(u)\}$$

Then, we define a Jaccard-`fsr` similarity value between u and $i \notin I^u$ as:

$$J\text{-}\texttt{fsr}(u,i) = \frac{|Rel^{u+} \cap F^i|}{|Rel^{u+} \cup F^i|} \tag{5.4}$$

5.3.6 Content Based- and Hybrid-`fsr`

Given the values of $s^{uf}(\cdot)$, we may compute a score associated to an item i not belonging to the original user profile by summing the values of relevant `fsr` in Rel^{u+} for those features belonging to the description of i. As we are adopting a pure content based approach we refer to this measure as Content-based feature spreading relevance (*CB*-`fsr`).

$$CB\text{-}\mathtt{fsr}(u,i) = \sum_{p \circ e \in Rel^{u+} \cap F^i} s^{uf}(p \circ e) \tag{5.5}$$

In the previous equation, as well as in Equation (5.4), we see that we consider only the relevant features in the users profile, or, in other words, the features they like.

In fact, this model makes use only of positive feedback. Even though this is fine-tuned feature-specific feedback, it still is positive feedback.
Instead, many other approaches take advantage of both positive and negative feedback. Some of them can rely on explicit negative feedback provided by users. Others adopt the *missing not at random* principle to set as negative all the unseen items. In this scenario, the situation is dramatically different, because the estimated rating is related to a feature, instead of an item.

To evaluate negative feedback for specific features, our idea is to exploit a Matrix Factorization (MF) model. These kinds of algorithms are often designed as iterative algorithms, with the aim of minimizing the rating prediction error. In a Matrix Factorization model, entities (users, and items) are represented by means of a bias and a vector of latent factors of predefined size D. In our scenario, we want to substitute items with features. For each user $u \in U$, and each feature $f \in F$ we build a binary vector $\mathbf{r^{uf}}(p \circ e) \in \mathbb{R}^{1 \times |F|}$, representing the estimated interaction between u and f. In this modeling, $\mathbf{r^{uf}}(p \circ e)$ contains only two 1 values corresponding to u and f if $f \in F^u$, while all the other values are set to 0. The prediction formula can be defined for each uf pair as:

$$r_H^{uf}(p \circ e) = w_0 + w_u + w_f + \sum_{k=1}^{D} v_{(u,k)} \cdot v_{(f,k)} \tag{5.6}$$

where the parameters to learn are: w_0 representing the global bias; w_u and w_f representing the individual biases for u, and f; the pair $v_{(u,k)}$ and $v_{(f,k)}$ in $\sum_{k=1}^{D} v_{(u,k)} \cdot v_{(f,k)}$ measuring the strength of the interaction between each pair u and f. The number of latent factors is represented by D. This value is usually selected at design time when implementing the MF algorithm.

Matrix Factorization can be easily trained in order to minimize the rating prediction error via gradient descent methods, alternating least-squares (ALS) and MCMC. Usually, when MF is optimized via gradient descent methods, the prediction formula can be defined as:

$$\delta^{uf} = w_0 + w_u + w_f + \sum_{k=1}^{D} v_{(u,k)} \cdot v_{(f,k)} \tag{5.7}$$

$$\hat{r}_H^{uf} = r_{min} + \frac{1}{1+e^{-\delta^{uf}}} \cdot range \tag{5.8}$$

where r_{min} is the minimum rating of all the considered transactions, whereas *range* is the rating range size. The rating prediction problem is bound to the $(0,1)$ range of the sigmoid function. The consequent cost function is:

$$J(\theta) \;=\; \sum_{u \in U} \sum_{f \in F^u} \left(r^{uf} - \hat{r}^{uf} \right) + \lambda_\Theta \cdot \|\Theta\|^2 \tag{5.9}$$

where r_H^{uf} is the feature rating estimated by our method, whereas $\hat{r}_H^{uf}$ is the rating estimated by the matrix factorization model. We can introduce $\sigma(\cdot)$ as a sigmoid function, and the update step can be defined as:

$$\begin{aligned} \Theta \;\leftarrow\; & \Theta + \alpha \cdot ((r^{uf} - \hat{r}^{uf}) \cdot \sigma(\delta^{uf}) \cdot (1 - \sigma(\delta^{uf})) \cdot \\ & \cdot range \cdot \frac{\partial}{\partial \Theta} \hat{r}^{uf} + \lambda \cdot \Theta) \end{aligned} \tag{5.10}$$

To update the factorized parameters, partial derivatives can be computed as:

$$\frac{\partial}{\partial \Theta} \hat{r}^{uf} = \begin{cases} 1, & \text{if } \theta = w_f, \\ v_{(u,k)}, & \text{if } \theta = v_{(f,k)}, \\ v_{(f,k)}, & \text{if } \theta = v_{(u,k)}, \\ 0, & \text{otherwise} \end{cases} \tag{5.11}$$

Using Equation (5.11) in Equation (5.10) the model parameters can be iteratively updated to minimize the rating prediction error.

As the resulting matrix contains both content-based and collaborative information (due to the Matrix Factorization), we refer to it as *hybrid profile*. To exploit it, we set the lowest K features (Rel^{u-}) as negative examples with a value $\hat{b}^{uf}$:

$$\hat{b}^{uf}(p \circ e) = -1 + \hat{s}^{uf}(p \circ e) \qquad \{p \circ e \in Rel^{u-}\} \tag{5.12}$$

where $\hat{s}^{uf}$ is the estimated feature f value for user u with the Matrix Factorization model. With both content-based, and collaborative filtering information we may compute a score associated to item i. We refer to this measure as *Hybrid-*`fsr`:

$$\begin{aligned} Hybrid\text{-}\mathtt{fsr}(u,i) &= \sum_{p \circ e \in Rel^{u+} \cap F^i} s^{uf}(p \circ e) \\ &+ \sum_{p \circ e \in Rel^{u-} \cap F^i} \hat{b}^{uf}(p \circ e) \end{aligned} \tag{5.13}$$

5.3.7 `frsCBF`

Values in $\mathscr{S}$ represent an indicator on the importance of a feature for a specific user. Based on this information we may compute recommendation lists for items $\bar{i} \notin I^u$ by exploiting $s^{uf}(p \circ e)$ values. We may compare features in F^u with those in $F^{\bar{i}}$ in different ways. The most intuitive way is that of computing a score for $\bar{i}$ is by summing of the values associated to the features it is composed by with a normalization factor.

$$\hat{r}^u(\bar{i}) = \frac{\sum_{p \circ e \in F^u \cap F^{\bar{i}}} s^{uf}(p \circ e)}{\sqrt{\sum_{p \circ e \in F^u \cap F^{\bar{i}}} (s^{uf}(p \circ e))^2}} \tag{5.14}$$

We can see that the value computed by Equation (5.14) corresponds to the cosine similarity between the user profile vector containing $s^{uf}(p \circ e)$ if $p \circ e \in F^u$ and 0 vice versa, and the corresponding binary vector for $\bar{i}$ containing 1 if $p \circ e \in F^i$ and 0 vice versa.

5.4 Experimental Evaluation

5.4.1 Experiments for Pure `FF`

In this section the experimental evaluation settings and the metrics used to evaluate the proposed algorithm are presented. We have evaluated the algorithms in terms of ranking accuracy for *top-N* recommendations. The evaluation has been carried out on two datasets, `LibraryThing` and `Last.fm` belonging respectively to the domains of books and music.

Datasets description and Pre-Processing

To alleviate the popularity bias from the evaluation results we have removed the 1% most popular items [69]. Moreover, we have removed users with a number of ratings smaller than five as we want to evaluate the algorithms in a non cold-start setting. The `LibraryThing` dataset contains $7,564$ users, $39,515$ items, and $797,299$ ratings. The minimum, mean and maximum number of ratings for user in the dataset are 20, 63, $3,018$, respectively. `Last.fm` contains $1,892$ users, $17,632$ items and $92,834$ ratings. In `LibraryThing`, ratings are distributed over a $1-10$ scale. In `Last.fm` the rating is the number of times a song has been played, hence that number has been rescaled for each user in a $1-10$ scale. Table 5.1 shows some statistics of the datasets subsets considering only the items mapped to `DBpedia` (using publicly available mappings [198]) after the pre-processing step. In case a mapping does not exist, a simple placeholder feature is used, that inherits the corresponding item values in terms of rating and relevance.

Table 5.1 also reports the sparsity values both for users-items and users-features matrices.

Evaluation protocol and experiment setting

To evaluate Pure `FF` we use the *all unrated items* [253] evaluation protocol, in which the ability to choose the correct set of items to propose to the users is favorite despite of the local ranking ability (*rated test-items* evaluation protocol). In

LibraryThing	# users	# items	# ratings	sparsity (%)
user-item space	6,909	12,656	248,589	99.7157
	# users	# features	# ratings	sparsity (%)
user-feature space	6,909	141,531	8,680,619	99.11226
Last.fm	# users	# items	# ratings	sparsity (%)
user-item space	1,866	8,502	39,557	99.75066
	# users	# features	# ratings	sparsity (%)
user-feature space	1,866	274,523	4,989,281	99.02603

Table 5.1: Datasets Statistics.

all unrated items the recommendation list is produced using as candidate list the Cartesian product between users and item minus the items the user experimented in the training set. Evaluation has been conducted using a hold-out $80-20$ splitting, in which 20% of the ratings are retained as test set. We have evaluated the accuracy of our approach by computing Precision ($P@N$), Recall ($R@N$), and nDCG ($nDCG@N$).

Baselines. In the experimental evaluation we have compared `FF` with the popularity baseline (PopRank) and, as we rely on Matrix Factorization, the well-known matrix factorization algorithm BPRMF [218] both in its pure collaborative version and in the hybrid one considering side information BPRMF+SI. We have also included PopRank as it is acknowledged that popularity ranking can show good performance and it is an important baseline to compare against [69]. To produce recommendation lists from these well-known algorithms we have used the *MyMediaLite*[1] implementation [104]. As for the selection of α and β parameters needed in Equation (5.2), in these experiments we have kept a conservative approach and we have set respectively α to the mean μ of the rated items, and β to the mean μ plus the standard deviation σ. Clearly, these values are not the optimal ones and the performance could be improved by a cross-validation setting of these parameters.

[1] `http://www.mymedialite.net/`

Experimental Results

Tables 5.2 and 5.3 show the performance of FF compared with the competing algorithms described in Section 5.4.1. In **bold** we mark the best result for each metric. All the evaluations have been performed by using the same protocols as implemented in RankSys[2] library [59].

In Table 5.2 we show the evaluation results on LibraryThing dataset with a threshold set to $7/10$ in a $Top - 10$ recommendation list. The ranking accuracy performance, measured through nDCG, Precision and Recall shows that Feature Factorization performs better than the competing algorithms. In details, Pure FF performs 4 to 6 times better than BPRMF, the second best accurate algorithm.

Alg	P@N	R@N	nDCG@N
Pure FF	**0.03251**	**0.06576**	**0.06129**
BPRMF	0.00837	0.01280	0.01020
BPRMF+SI	0.00777	0.01325	0.01007
PopRank	0.00023	0.00095	0.00044

Table 5.2: Comparative results on LibraryThing dataset, Top-10 recommendation list and relevance threshold of 7/10.

As the rescaling operation in Last.fm affects the values of the items in the test set, we have decided to evaluate considering all the items in test set as relevant (i.e., setting the relevance threshold to 0). Table 5.3 shows ranking accuracy evaluation results on Last.fm dataset with a threshold of $0/10$ for a $Top - 10$ recommendation list. For precision metric the best performing algorithm is Pure FF that performs 4 times better than BPRMF. For nDCG, Feature Factorization performs at least 5 times better than the competing algorithms. The differences about accuracy metrics between Pure FF and the other algorithms are statistically significant according to the Student's paired *t-test* with $p < 0.001$ for every cases.

[2] https://github.com/RankSys/RankSys

Alg	P@N	R@N	nDCG@N
Pure FF	**0.01543**	**0.02701**	**0.02330**
BPRMF	0.00348	0.00902	0.00495
BPRMF+SI	0.00032	0.00073	0.00028
PopRank	0.00027	0.00089	0.00021

Table 5.3: Comparative results on `Last.fm` dataset, Top-10 recommendation list and no relevance threshold.

5.4.2 Experimental Evaluation for FF extensions

The evaluation has been carried out on three well-known datasets, `LibraryThing`, `MovieLens`, and `Last.fm` belonging respectively to the domains of books, movies, and music.

Datasets description and Pre-Processing

Again, to alleviate the popularity bias from the evaluation results, we have removed the 1% most popular items [69]. Moreover, we have removed users with a number of ratings smaller than five as we want to evaluate the algorithms in a non-cold-start setting. The `MovieLens` dataset contains $1,000,209$ ratings over of approximately $3,900$ movies made by $6,040$*users* on the Movielens platform. Each rating is expressed on a $1-5$ scale. The `LibraryThing` dataset contains $7,564$ users, $39,515$ items and $797,299$ ratings. The minimum, mean and maximum number of ratings per user in the dataset are 20, 63 and $3,018$, respectively. `Last.fm` contains $1,892$ users, $17,632$ items and $92,834$ ratings. In `LibraryThing`, ratings are distributed over a $1-10$ scale. In `Last.fm` the rating is the number of times a song has been played, hence that number has been rescaled for each user on a $1-10$ scale. Table 5.4 shows some statistics of the datasets subsets considering only the items mapped to `DBpedia` (using publicly available mappings [198]) after the pre-processing step. Table 5.4 also reports the sparsity values both for `Users-Items` and Users-Features matrices.

`LibraryThing`	# users	# items	# ratings	sparsity (%)
user-item space	6,909	12,656	248,589	99.7157
	# users	# features	# ratings	sparsity (%)
user-feature space	6,909	141,531	8,680,619	99.11226
`Last.fm`	# users	# items	# ratings	sparsity (%)
user-item space	1,866	8,502	39,557	99.75066
	# users	# features	# ratings	sparsity (%)
user-feature space	1,866	274,523	4,989,281	99.02603
`MovieLens`	# users	# items	# ratings	sparsity (%)
user-item space	6,040	3,171	689,867	96.3981
	# users	# features	# ratings	sparsity (%)
user-feature space	6,040	100,845	56,732,878	90.68584

Table 5.4: Datasets Statistics.

Evaluation protocol and experiment setting

To evaluate `FF`, even here, we adopt the *all unrated items* [253] evaluation protocol. In *all unrated items* the recommendation list is produced using as candidates list the Cartesian product between users and item minus the items the user experimented in the training set. The evaluation has been conducted using a hold-out $80-20$ splitting, in which 20% of the ratings are retained as the test set.

We chose to evaluate the approaches through accuracy, novelty and aggregate diversity metrics. The *top-N* recommendation accuracy metrics we used are Precision ($P@N$), Recall($R@N$) and nDCG ($nDCG@N$). As for novelty, in the last years, several metrics have been proposed, which measure the ability of a system to recommend items that are not very popular within the catalog and that usually belong to the long tail [147]. In this investigation, *EPC* (Expected Popularity Complement) is used for novelty evaluation. *EPC* measures how much a system is capable to propose long tail items to the users. In this case, the binary relevance formulation of *EPC* has been adopted [59]. Diversity has been measured through *catalog coverage* (aggregate diversity in *top-N* list), *Gini index* and *Shannon entropy*. The *catalog coverage*, also called aggregate diversity or the diversity-in-top-N ($ADiv@N$), formulated in [5], denotes the overall number of different items recommended within all recommendation lists. It denotes the propensity of a system to recommend always the same items. *Gini index* (*Gini*) and Shannon entropy (*SE*) are two different metrics used to measure the distributional inequality [59]. The evaluation has been performed considering $Top-5$ and $Top-10$ recommendations for `MovieLens`, `Last.fm` and `LibraryThing` datasets and two different thresholds for deeming items as relevant: 7 and 9. Accuracy and novelty metrics have been computed on a per-user basis and the results have been averaged.

Baselines

In the experimental evaluation, we have compared `FF` with the popularity baseline and three well-known Matrix Factorization algorithms. These latter have been tested both in their pure-collaborative variant and in their hybrid ones by also considering side information coming with the dataset (Tag), i.e., tags associated to the items, or considering `Linked Data` side information extracted from `DBpedia` (`LOD`). Hybrid variants of matrix factorization algorithms have been realized feeding them with new transactions corresponding to items-features associations, as suggested in [186]. The competing algorithms we selected are then:

- `PopRank`, a non-personalized algorithm that produces the same recommendation list for all the users. This list is computed measuring the items' popularity and ordering them in descending order. It is acknowledged that popularity ranking can show good performance and it is an important baseline to compare against [69].

- `Bayesian personalized ranking - Matrix Factorization` (BPRMF), a matrix factorization algorithm that exploits the Bayesian Personalized Ranking criterion[218] to minimize the ranking errors.

- `Soft Margin Ranking - Matrix Factorization` (SMRMF), a matrix factorization model that uses stochastic gradient descent to optimize a soft margin [277].

- Weighted Matrix Factorization (WRMF) a weighted matrix factorization model that makes use of alternate least squares [130] for optimization.

To produce recommendation lists from these well-known algorithms we have used the *MyMediaLite*[3] implementation [104]. The experimental setting is designed to evaluate the performance of the proposed approach against different matrix factorization algorithms, varying datasets, and side-information fed to the recommender, keeping fixed the models' hyperparameters. For this reason, the parameters are set considering the values suggested by the authors in their original papers.

Experimental Results

Tables 5.5 to 5.18 show the performance of FF[4] compared with the competing algorithms described in Section 5.4.2. In **bold** we mark the best result for each metric while we underline the second best result. The differences about accuracy metrics between FF and the other algorithms are statistically significant according to the Student's paired *t-test* [250] with $p \ll 0.001$.

Ranking accuracy evaluation The first three metrics in Tables 5.5 to 5.18 are devoted to accuracy evaluation of FF with respect to competing baselines. As it could be appreciated in each experiment at least one variant of FF outperforms the baselines. However, it should be noticed that, with respect to results from the literature, the removal of the 1% heavily affects the collaborative algorithms. For this reason, it is important to underline that Hybrid-fsr also suffers from this removal. It is worth to notice that this performance drop is alleviated by its hybrid nature. Despite this consideration, collaborative algorithms in MovieLens experiments still show high performance, with BPRMF as the winner among the competitors. Nevertheless, LibraryThing and Last.fm showed that the algorithm that best

[3] http://www.mymedialite.net/

[4] Implementation available at: https://github.com/sisinflab/Features-Factorization

deals with increasing sparsity is WRMF. Among the proposed variants the discussion is more complex because from experiments it is clear that they behave as they were completely different algorithms. In details, there could be noticed two different dimensions for the results analysis: quality of meta-data, and statistics of datasets. The latter seems to impose the magnitude of the metrics value, which is comparable to competing algorithms. The former heavily affects the results and we consider it so important to be detailed in the next subsection to explain the details of the measured differences.

RQ1: Are all the retrieved features needed to get results comparable with state-of-the-art algorithms?

This question is motivated by a rich literature in feature selection research field. In our previous experiment (regarding Pure `FF`), we have decided to take advantage of all available features. However, since it was a collaborative filtering algorithm, we have found really hard to understand if all the features have been effectively beneficial. There have been too many parameters that could affect the results, and too many assumptions to be made. To establish whether they have been beneficial or not, we have decided to define a simple a clear content-based variant of `FF` based on cosine similarity and compare it against a Jaccard-based variant. In this variant, a threshold been defined to consider only relevant features. The choice of the threshold could be unfair, thus we have decided to adopt the mean of features values as a reasonable threshold (see Equation (5.3)). The experiments clearly show that the performance of the two variants depends on the specific dataset and items' descriptions. In `MovieLens` we can observe rich descriptions for almost all the considered items, it is therefore not surprising that the winner is CB-`fsr`, in which all features are involved. `LibraryThing` shows that the winner is Jaccard-`fsr` Tag. This leads to two considerations: 1) item descriptions in KB are not as effective (for recommendation task) as dataset tags are; 2) Among the considered tags, performing a feature selection is beneficial. Even in `Last.fm`, `LOD` features are worse than tags, and it could be noticed that items descriptions may vary from very rich ones to items with less than 5 features.

RQ2: How much does FF performance depend on the number of items we are able to find a mapping for?

This is a very common question that is usually posed when considering the injection of `Linked Open Data` in a recommendation scenario. Despite the obvious fact that collecting high-quality users-based tags using crowdsourcing may result in a difficult task, we have decided to perform a series of specific experiments to answer the question. To test how much the lack of some items affects the overall recommendation results, we have dropped half of the Hybrid-`fsr` items repeatedly until we have reached one eight of the original number of items. It is possible to appreciate the number of considered items in the first column of each table, near each Hybrid-`fsr` experiment (e.g., Hybrid-`fsr` 3,171, 1,586, 793, 396 for `MovieLens`). For each case, the items have been dropped with uniform distribution, and the model has been completely retrained. In `MovieLens`, the performance of Hybrid-`fsr` with one-half items shows results that are still comparable against the competing collaborative filtering algorithms. Moreover, the results of one-eight items experiments show that the decrease w.r.t. original performance is only of two-thirds of the original value. In `LibraryThing`, the decrease is more evident, but its hybrid nature makes Hybrid-`fsr` still comparable with other algorithms even though the number of considered items is reduced to one eight of the original number. In `Last.fm`, the decrease in performance is even stronger. However, this may be due to the specific dataset, because the same behavior can be observed on all the baselines, but WRMF.

RQ3: Is it possible to reduce the computational effort of FF and keep at least the same results in terms of accuracy?

The adoption of a model that takes into account matrix factorization in a much bigger matrix (Users-Features matrix) could be considered a questionable option in very big datasets. For this reason, we have decided to create the pure content-based variants of `FF`. Since we have created two different content-based variants for the aforementioned reasons, we consider useful to discuss the results of all the three variants together. Moreover, since Hybrid-`fsr` has been designed to work mainly with LOD side-information, we consider the LOD versions of the variants. In MovieLens we may notice rich descriptions and highly collaboratively-connected users and items, thus it is reasonable that the winner is CB-fsr, followed by Hybrid-fsr. In LibraryThing and Last.fm, the ranking order is different, and Hybrid-fsr is the winner in these cases. These behaviors are reasonably derived from the quality of descriptions: with worse descriptions, CB-fsr is the last of the ranked list. However, it is noteworthy that with less-described items, Hybrid-fsr is able to exploit collaborative information and it performs better than the others.

RQ4: What is the influence of the quality of Linked Data information on the recommendation results?

We have mentioned the differences in terms of performance in previous discussions, however, this is another common question related to the adoption of Linked Open Data. In order to answer it, we have decided to feed all the available algorithms with both: LOD, and Tag side-information. In MovieLens all the FF variants work better with LOD information than tags. For the competing approaches, WRMF seems to be the only one in which performance with LOD is better than its other variants. In LibraryThing, we can observe the opposite behavior, with Jaccard-fsr Tag as a winner, and this means that tags are better than LOD for item recommendation. However, we can observe that the winning of Jaccard variant means that almost half of these features have limited importance for the recommendation task. The consideration we have made before about not homogeneous Last.fm LOD description is confirmed by the performance of Tag versions against LOD ones.

RQ5: How does FF performs with reference to diversity and novelty of recommendations?

Even though accuracy is really important to establish the quality of a recommender system, when different variants are proposed, an evaluation of the impact on novelty and diversity of recommendation is mandatory. Novelty and diversity metrics have

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-`fsr` LOD	0.06328	<u>0.02982</u>	0.05783	0.05970	804	0.03454	6.82679
CB-`fsr` LOD	**0.08563**	**0.03593**	**0.07435**	**0.07903**	552	0.01935	5.98567
Jaccard-`fsr` Tag	0.02238	0.00939	0.01578	0.02016	591	0.03404	7.06170
CB-`fsr` Tag	0.02377	0.00790	0.01563	0.02127	599	0.05414	7.79185
Hybrid-`fsr` 3171	<u>0.06411</u>	0.02965	<u>0.05831</u>	<u>0.06061</u>	823	0.03519	6.84044
Hybrid-`fsr` 1586	0.05593	0.01997	0.04292	0.04913	718	0.03970	7.08804
Hybrid-`fsr` 793	0.05066	0.01547	0.03817	0.04651	527	0.03629	7.13862
Hybrid-`fsr` 396	0.02321	0.00466	0.01759	0.02641	298	0.02500	6.66303
BPRMF	0.05808	0.02494	0.04548	0.04934	**1109**	**0.09286**	**8.54816**
BPRMF Tag	0.05801	0.02564	0.04669	0.04944	<u>1069</u>	<u>0.08744</u>	<u>8.43765</u>
BPRMF LOD	0.04394	0.00972	0.03588	0.04050	772	0.03234	6.70888
SMRMF	0.03828	0.01635	0.02838	0.03375	612	0.03670	7.22453
SMRMF Tag	0.04152	0.01830	0.03222	0.03802	621	0.04212	7.41911
SMRMF LOD	0.02454	0.00877	0.01852	0.02270	481	0.02262	6.49055
WRMF	0.02950	0.02250	0.02786	0.02344	535	0.04192	7.47313
WRMF Tag	0.03411	0.02447	0.03119	0.02727	532	0.04000	7.39674
WRMF LOD	0.03404	0.02473	0.03139	0.02695	513	0.03841	7.34936
MostPopular	0.00970	0.00431	0.00764	0.00683	51	0.00240	3.47028

Table 5.5: Comparative results on `MovieLens` dataset, Top-5 recommendation list and relevance threshold of 4/5.

been measured for all the considered experiments, and we suddenly may notice that recommendation lists of Hybrid-`fsr` show to be more diversified w.r.t. CB-`fsr`. This may due to the nature of the proposed hybrid algorithm, that makes use of novel features to perform recommendations. Moreover, it could be noticed that about CB-`fsr` the decrease in diversity seems to be related to the increase in accuracy metrics. If we look at the literature, this is probably due to the overspecialization issue that affects content-based algorithms. In this evaluation, these considerations are coherent to results on `LibraryThing`, and `Last.fm` datasets. For the sake of completeness, it is important to underline that the absolute winner in terms of aggregate diversity and distributional inequality on `MovieLens`, and `Last.fm` is BPRMF. However, its low performance in terms of accuracy makes us not consider it in this discussion. Finally, Novelty results behave in a coherent way along with accuracy metrics, and this denotes that performance registered for popular items are confirmed on the long tail.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-`fsr` LOD	0.05801	0.04868	0.06291	0.05509	1065	0.04348	7.25795
CB-`fsr` LOD	**0.07363**	**0.05875**	**0.07810**	**0.07044**	736	0.02571	6.47117
Jaccard-`fsr` Tag	0.02071	0.01725	0.01820	0.01936	853	0.04949	7.61902
CB-`fsr` Tag	0.02174	0.01510	0.01774	0.02016	845	0.07750	8.32482
Hybrid-`fsr` 3171	0.05838	0.04882	0.06334	0.05568	1093	0.04457	7.28407
Hybrid-`fsr` 1586	0.05240	0.03500	0.04681	0.04723	890	0.05024	7.53621
Hybrid-`fsr` 793	0.04627	0.02575	0.03969	0.04368	620	0.04512	7.53465
Hybrid-`fsr` 396	0.01732	0.00684	0.01511	0.02088	339	0.03069	7.02743
BPRMF	0.05356	0.04640	0.05169	0.04698	**1387**	**0.11563**	**8.88650**
BPRMF Tag	0.05430	0.04829	0.05347	0.04733	1358	0.10926	8.78973
BPRMF LOD	0.03672	0.01830	0.03413	0.03511	1113	0.04512	7.30863
SMRMF	0.03619	0.02958	0.03269	0.03238	905	0.05203	7.72057
SMRMF Tag	0.03854	0.03349	0.03678	0.03568	857	0.05725	7.88452
SMRMF LOD	0.02169	0.01487	0.01952	0.02067	731	0.03263	7.00065
WRMF	0.03015	0.04300	0.03639	0.02416	663	0.05506	7.88662
WRMF Tag	0.03276	0.04592	0.03937	0.02684	670	0.05327	7.82256
WRMF LOD	0.03369	0.04630	0.04015	0.02719	635	0.05073	7.76383
MostPopular	0.01235	0.01399	0.01189	0.00838	83	0.00475	4.35932

Table 5.6: Comparative results on `MovieLens` dataset, Top-10 recommendation list and relevance threshold of 4/5.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-`fsr` LOD	0.07430	0.02487	0.06133	0.06975	804	0.03454	6.82679
CB-`fsr` LOD	**0.10020**	**0.02993**	**0.07913**	**0.09172**	552	0.01935	5.98567
Jaccard-`fsr` Tag	0.03513	0.00958	0.01897	0.03140	591	0.03404	7.06170
CB-`fsr` Tag	0.03225	0.00753	0.01821	0.02873	599	0.05414	7.79185
Hybrid-`fsr` 3171	0.07560	0.02487	0.06194	0.07099	823	0.03519	6.84044
Hybrid-`fsr` 1586	0.06868	0.01753	0.04700	0.06030	718	0.03970	7.08804
Hybrid-`fsr` 793	0.06497	0.01379	0.04296	0.05964	527	0.03629	7.13862
Hybrid-`fsr` 396	0.03341	0.00441	0.02093	0.03728	298	0.02500	6.66303
BPRMF	0.07295	0.02167	0.05050	0.06216	**1109**	**0.09286**	**8.54816**
BPRMF Tag	0.07215	0.02233	0.05126	0.06176	1069	0.08744	8.43765
BPRMF LOD	0.05384	0.00857	0.03938	0.04944	772	0.03234	6.70888
SMRMF	0.05179	0.01512	0.03253	0.04582	612	0.03670	7.22453
SMRMF Tag	0.05632	0.01767	0.03703	0.05182	621	0.04212	7.41911
SMRMF LOD	0.03152	0.00773	0.02082	0.02906	481	0.02262	6.49055
WRMF	0.03675	0.01998	0.02977	0.02926	535	0.04192	7.47313
WRMF Tag	0.04185	0.02130	0.03328	0.03323	532	0.04000	7.39674
WRMF LOD	0.04152	0.02194	0.03351	0.03284	513	0.03841	7.34936
MostPopular	0.01179	0.00355	0.00814	0.00825	51	0.00240	3.47028

Table 5.7: Comparative results on `MovieLens` dataset, Top-5 recommendation list and relevance threshold of 3/5.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-`fsr` LOD	0.06980	0.04184	0.06590	0.06554	1065	0.04348	7.25795
CB-`fsr` LOD	**0.08791**	**0.04943**	**0.08197**	**0.08312**	736	0.02571	6.47117
Jaccard-`fsr` Tag	0.03180	0.01664	0.02117	0.02980	853	0.04949	7.61902
CB-`fsr` Tag	0.02957	0.01426	0.02016	0.02736	845	0.07750	8.32482
Hybrid-`fsr` 3171	0.07020	0.04179	0.06630	0.06621	1093	0.04457	7.28407
Hybrid-`fsr` 1586	0.06573	0.03091	0.05061	0.05891	890	0.05024	7.53621
Hybrid-`fsr` 793	0.06078	0.02312	0.04415	0.05695	620	0.04512	7.53465
Hybrid-`fsr` 396	0.02533	0.00654	0.01787	0.02986	339	0.03069	7.02743
BPRMF	0.06770	0.04054	0.05604	0.05944	**1387**	**0.11563**	**8.88650**
BPRMF Tag	0.06950	0.04302	0.05773	0.06031	1358	0.10926	8.78973
BPRMF LOD	0.04551	0.01605	0.03723	0.04325	1113	0.04512	7.30863
SMRMF	0.04940	0.02766	0.03659	0.04426	905	0.05203	7.72057
SMRMF Tag	0.05224	0.03213	0.04122	0.04859	857	0.05725	7.88452
SMRMF LOD	0.02803	0.01341	0.02167	0.02659	731	0.03263	7.00065
WRMF	0.03795	0.03892	0.03812	0.03040	663	0.05506	7.88662
WRMF Tag	0.04066	0.04082	0.04118	0.03306	670	0.05327	7.82256
WRMF LOD	0.04190	0.04124	0.04201	0.03365	635	0.05073	7.76383
MostPopular	0.01581	0.01206	0.01247	0.01058	83	0.00475	4.35932

Table 5.8: Comparative results on `MovieLens` dataset, Top-10 recommendation list and relevance threshold of 3/5.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-`fsr` LOD	0.02822	0.04786	0.04560	0.03203	4043	0.09912	10.47816
CB-`fsr` LOD	0.02921	0.04786	0.04908	0.03478	3825	0.07815	9.88738
Jaccard-`fsr` Tag	**0.04009**	**0.07187**	**0.07124**	**0.04863**	2989	0.05186	9.30556
CB-`fsr` Tag	0.03763	0.06848	0.06936	0.04570	1519	0.01962	7.92150
Hybrid-`fsr` 13073	0.02866	0.04836	0.04593	0.03237	**4098**	**0.10247**	**10.53733**
Hybrid-`fsr` 6851	0.01719	0.02409	0.02720	0.02072	3127	0.06968	9.87870
Hybrid-`fsr` 3425	0.00978	0.01306	0.01623	0.01255	2017	0.04219	8.99890
Hybrid-`fsr` 1712	0.00423	0.00487	0.00687	0.00557	1156	0.02221	8.17881
BPRMF	0.00481	0.00686	0.00643	0.00490	1604	0.03418	9.11855
BPRMF Tag	0.00417	0.00680	0.00607	0.00430	2317	0.04500	9.35826
BPRMF LOD	0.00090	0.00102	0.00132	0.00103	1310	0.01386	7.14172
SMRMF	0.00347	0.00506	0.00468	0.00369	1202	0.01937	8.08369
SMRMF Tag	0.00249	0.00345	0.00313	0.00253	1089	0.01164	7.10852
SMRMF LOD	0.00127	0.00153	0.00185	0.00146	1261	0.01503	7.36249
WRMF	0.00625	0.01163	0.01001	0.00643	370	0.00771	7.06538
WRMF Tag	0.00454	0.00837	0.00686	0.00451	619	0.00864	7.11308
WRMF LOD	0.00321	0.00626	0.00485	0.00314	359	0.00482	6.38107
MostPopular	0.00012	0.00025	0.00011	0.00008	12	0.00033	2.46979

Table 5.9: Comparative results on `LibraryThing` dataset, Top-5 recommendation list and relevance threshold of 9/10.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-`fsr` LOD	0.01966	0.06244	0.05009	0.02468	5213	0.11331	10.69759
CB-`fsr` LOD	0.01944	0.06220	0.05273	0.02598	5192	0.09369	10.18603
Jaccard-`fsr` Tag	**0.02723**	**0.09652**	**0.07852**	**0.03661**	4038	0.05882	9.51908
CB-`fsr` Tag	0.02435	0.08563	0.07396	0.03352	2011	0.02213	8.20660
Hybrid-`fsr` 13073	0.01987	0.06265	0.05035	0.02491	**5274**	**0.11708**	**10.75585**
Hybrid-`fsr` 6851	0.01087	0.02992	0.02781	0.01504	3907	0.07767	10.09293
Hybrid-`fsr` 3425	0.00575	0.01512	0.01573	0.00875	2357	0.04635	9.24221
Hybrid-`fsr` 1712	0.00265	0.00587	0.00660	0.00399	1337	0.02543	8.50527
BPRMF	0.00408	0.01163	0.00802	0.00435	2187	0.04422	9.50059
BPRMF Tag	0.00389	0.01247	0.00822	0.00405	3263	0.05861	9.76858
BPRMF LOD	0.00069	0.00184	0.00152	0.00084	1878	0.01721	7.46591
SMRMF	0.00310	0.00937	0.00622	0.00333	1809	0.02768	8.65574
SMRMF Tag	0.00285	0.00837	0.00512	0.00276	1686	0.01784	7.79521
SMRMF LOD	0.00096	0.00248	0.00210	0.00117	1942	0.02260	8.04500
WRMF	0.00576	0.02056	0.01351	0.00599	514	0.01181	7.67982
WRMF Tag	0.00434	0.01583	0.00977	0.00437	935	0.01274	7.69802
WRMF LOD	0.00358	0.01295	0.00754	0.00340	584	0.00753	7.02718
MostPopular	0.00019	0.00101	0.00040	0.00014	20	0.00075	3.57568

Table 5.10: Comparative results on `LibraryThing` dataset, Top-10 recommendation list and relevance threshold of 9/10.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-`fsr` LOD	0.05853	0.06017	0.06312	0.06646	4043	0.09912	10.47816
CB-`fsr` LOD	0.05182	0.05403	0.06314	0.06052	3825	0.07815	9.88738
Jaccard-`fsr` Tag	**0.07735**	**0.08662**	**0.09597**	**0.09279**	2989	0.05186	9.30556
CB-`fsr` Tag	0.06195	0.06839	0.08517	0.07385	1519	0.01962	7.92150
Hybrid-`fsr` 13073	0.05931	0.06053	0.06332	0.06722	**4098**	**0.10247**	**10.53733**
Hybrid-`fsr` 6851	0.03671	0.02982	0.03702	0.04334	3127	0.06968	9.87870
Hybrid-`fsr` 3425	0.02009	0.01583	0.02194	0.02549	2017	0.04219	8.99890
Hybrid-`fsr` 1712	0.00932	0.00579	0.00937	0.01198	1156	0.02221	8.17881
BPRMF	0.00926	0.00702	0.00814	0.00938	1604	0.03418	9.11855
BPRMF Tag	0.00857	0.00723	0.00760	0.00849	2317	0.04500	9.35826
BPRMF LOD	0.00165	0.00104	0.00158	0.00175	1310	0.01386	7.14172
SMRMF	0.00674	0.00455	0.00569	0.00713	1202	0.01937	8.08369
SMRMF Tag	0.00524	0.00374	0.00440	0.00559	1089	0.01164	7.10852
SMRMF LOD	0.00240	0.00198	0.00259	0.00267	1261	0.01503	7.36249
WRMF	0.01366	0.01679	0.01515	0.01421	370	0.00771	7.06538
WRMF Tag	0.00964	0.01116	0.00958	0.00963	619	0.00864	7.11308
WRMF LOD	0.00701	0.00846	0.00726	0.00686	359	0.00482	6.38107
MostPopular	0.00012	0.00016	0.00011	0.00008	12	0.00033	2.46979

Table 5.11: Comparative results on `LibraryThing` dataset, Top-5 recommendation list and relevance threshold of 7/10.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-`fsr` LOD	0.04183	0.07904	0.06855	0.05195	<u>5213</u>	<u>0.11331</u>	<u>10.69759</u>
CB-`fsr` LOD	0.03484	0.06922	0.06678	0.04559	5192	0.09369	10.18603
Jaccard-`fsr` Tag	**0.05303**	**0.11633**	**0.10434**	**0.07030**	4038	0.05882	9.51908
CB-`fsr` Tag	<u>0.04148</u>	<u>0.08711</u>	<u>0.08966</u>	<u>0.05526</u>	2011	0.02213	8.20660
Hybrid-`fsr` 13073	0.04247	0.07893	0.06871	0.05262	**5274**	**0.11708**	**10.75585**
Hybrid-`fsr` 6851	0.02390	0.03677	0.03742	0.03204	3907	0.07767	10.09293
Hybrid-`fsr` 3425	0.01236	0.01851	0.02105	0.01818	2357	0.04635	9.24221
Hybrid-`fsr` 1712	0.00592	0.00710	0.00883	0.00867	1337	0.02543	8.50527
BPRMF	0.00837	0.01280	0.01020	0.00869	2187	0.04422	9.50059
BPRMF Tag	0.00777	0.01325	0.01007	0.00793	3263	0.05861	9.76858
BPRMF LOD	0.00127	0.00166	0.00172	0.00145	1878	0.01721	7.46591
SMRMF	0.00592	0.00836	0.00718	0.00638	1809	0.02768	8.65574
SMRMF Tag	0.00556	0.00835	0.00653	0.00568	1686	0.01784	7.79521
SMRMF LOD	0.00191	0.00315	0.00287	0.00223	1942	0.02260	8.04500
WRMF	0.01253	0.02829	0.01982	0.01315	514	0.01181	7.67982
WRMF Tag	0.00909	0.02006	0.01333	0.00921	935	0.01274	7.69802
WRMF LOD	0.00763	0.01679	0.01080	0.00729	584	0.00753	7.02718
MostPopular	0.00023	0.00095	0.00044	0.00017	20	0.00075	3.57568

Table 5.12: Comparative results on `LibraryThing` dataset, Top-10 recommendation list and relevance threshold of 7/10.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-`fsr` LOD	0.00482	0.01943	0.01561	0.00568	899	0.02222	7.88347
CB-`fsr` LOD	0.00397	0.01679	0.01185	0.00451	772	0.01347	6.77499
Jaccard-`fsr` Tag	<u>0.00547</u>	<u>0.02206</u>	<u>0.01705</u>	<u>0.00653</u>	<u>1058</u>	<u>0.03078</u>	<u>8.45781</u>
CB-`fsr` Tag	**0.00740**	**0.03135**	**0.02356**	**0.00851**	1008	0.02621	8.07217
Hybrid-`fsr` 10004	0.00514	0.02041	0.01612	0.00602	946	0.02401	8.01508
Hybrid-`fsr` 5002	0.00311	0.01215	0.00841	0.00341	1028	**0.03232**	**8.57696**
Hybrid-`fsr` 2051	0.00118	0.00482	0.00320	0.00143	907	0.02905	8.44240
Hybrid-`fsr` 1250	0.00032	0.00134	0.00101	0.00034	656	0.02387	8.22125
BPRMF	0.00032	0.00080	0.00047	0.00025	253	0.00599	6.28848
BPRMF Tag	0.00000	0.00000	0.00000	0.00000	151	0.00099	3.43110
BPRMF LOD	0.00000	0.00000	0.00000	0.00000	**1112**	0.02934	8.14561
SMRMF	0.00011	0.00054	0.00034	0.00011	364	0.00647	6.24558
SMRMF Tag	0.00011	0.00018	0.00000	0.00007	411	0.00648	5.99848
SMRMF LOD	0.00000	0.00000	0.00000	0.00000	777	0.01832	7.51538
WRMF	0.00236	0.00857	0.00491	0.00208	186	0.00839	6.79522
WRMF Tag	0.00171	0.00589	0.00393	0.00160	300	0.00820	6.71812
WRMF LOD	0.00171	0.00679	0.00351	0.00170	334	0.00829	6.75719
MostPopular	0.00000	0.00000	0.00000	0.00000	9	0.00043	2.50907

Table 5.13: Comparative results on `Last.fm` dataset, Top-5 recommendation list and relevance threshold of 9/10.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-fsr LOD	0.00311	0.02479	0.01750	0.00417	1254	0.02848	8.24498
CB-fsr LOD	0.00284	0.02291	0.01440	0.00354	1196	0.01932	7.34914
Jaccard-fsr Tag	0.00402	0.03224	0.02012	0.00512	1544	**0.03931**	8.80789
CB-fsr Tag	**0.00456**	**0.03760**	**0.02570**	**0.00613**	1519	0.03539	8.52017
Hybrid-fsr 10004	0.00327	0.02568	0.01803	0.00440	1325	0.03072	8.38414
Hybrid-fsr 5002	0.00193	0.01563	0.00955	0.00246	1421	0.03900	**8.85578**
Hybrid-fsr 2051	0.00086	0.00674	0.00384	0.00112	1191	0.03486	8.71436
Hybrid-fsr 1250	0.00048	0.00402	0.00190	0.00045	807	0.02910	8.54335
BPRMF	0.00027	0.00134	0.00061	0.00023	334	0.00768	6.66212
BPRMF Tag	0.00000	0.00000	0.00000	0.00000	242	0.00226	4.75911
BPRMF LOD	0.00000	0.00000	0.00000	0.00000	**1591**	0.03678	8.45905
SMRMF	0.00016	0.00161	0.00069	0.00015	554	0.00952	6.75383
SMRMF Tag	0.00016	0.00098	0.00021	0.00012	652	0.01040	6.72105
SMRMF LOD	0.00000	0.00000	0.00000	0.00000	1234	0.02770	8.17090
WRMF	0.00188	0.01358	0.00654	0.00183	257	0.01195	7.28991
WRMF Tag	0.00150	0.01063	0.00567	0.00149	452	0.01346	7.45752
WRMF LOD	0.00150	0.01179	0.00529	0.00155	552	0.01376	7.48693
MostPopular	0.00005	0.00018	0.00000	0.00004	14	0.00097	3.48598

Table 5.14: Comparative results on `Last.fm` dataset, Top-10 recommendation list and relevance threshold of 9/10.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-fsr LOD	0.00675	0.02364	0.01996	0.00792	899	0.02222	7.88347
CB-fsr LOD	0.00547	0.02108	0.01464	0.00603	772	0.01347	6.77499
Jaccard-fsr Tag	0.00868	0.03131	0.02330	0.00971	1058	0.03078	8.45781
CB-fsr Tag	**0.01115**	**0.04118**	**0.03115**	**0.01257**	1008	0.02621	8.07217
Hybrid-fsr 10004	0.00707	0.02449	0.02057	0.00829	946	0.02401	8.01508
Hybrid-fsr 5002	0.00482	0.01657	0.01258	0.00571	1028	**0.03232**	**8.57696**
Hybrid-fsr 2051	0.00161	0.00634	0.00427	0.00195	907	0.02905	8.44240
Hybrid-fsr 1250	0.00075	0.00259	0.00181	0.00082	656	0.02387	8.22125
BPRMF	0.00054	0.00143	0.00081	0.00045	253	0.00599	6.28848
BPRMF Tag	0.00000	0.00000	0.00000	0.00000	151	0.00099	3.43110
BPRMF LOD	0.00011	0.00018	0.00004	0.00008	**1112**	0.02934	8.14561
SMRMF	0.00032	0.00161	0.00078	0.00025	364	0.00647	6.24558
SMRMF Tag	0.00011	0.00013	0.00000	0.00007	411	0.00648	5.99848
SMRMF LOD	0.00011	0.00018	0.00004	0.00009	777	0.01832	7.51538
WRMF	0.00397	0.01456	0.00920	0.00385	186	0.00839	6.79522
WRMF Tag	0.00279	0.00987	0.00627	0.00259	300	0.00820	6.71812
WRMF LOD	0.00236	0.00831	0.00493	0.00230	334	0.00829	6.75719
MostPopular	0.00000	0.00000	0.00000	0.00000	9	0.00043	2.50907

Table 5.15: Comparative results on `Last.fm` dataset, Top-5 recommendation list and relevance threshold of 7/10.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-`fsr` LOD	0.00477	0.03391	0.02333	0.00610	1254	0.02848	8.24498
CB-`fsr` LOD	0.00391	0.02873	0.01786	0.00476	1196	0.01932	7.34914
Jaccard-`fsr` Tag	<u>0.00665</u>	<u>0.04586</u>	<u>0.02774</u>	<u>0.00791</u>	<u>1544</u>	**0.03931**	<u>8.80789</u>
CB-`fsr` Tag	**0.00734**	**0.05332**	**0.03524**	**0.00943**	1519	0.03539	8.52017
Hybrid-`fsr` 10004	0.00482	0.03356	0.02361	0.00629	1325	0.03072	8.38414
Hybrid-`fsr` 5002	0.00327	0.02282	0.01465	0.00430	1421	<u>0.03900</u>	**8.85578**
Hybrid-`fsr` 2051	0.00123	0.00984	0.00543	0.00158	1191	0.03486	8.71436
Hybrid-`fsr` 1250	0.00070	0.00482	0.00270	0.00076	807	0.02910	8.54335
BPRMF	0.00043	0.00214	0.00095	0.00040	334	0.00768	6.66212
BPRMF Tag	0.00000	0.00000	0.00000	0.00000	242	0.00226	4.75911
BPRMF LOD	0.00011	0.00045	0.00012	0.00008	**1591**	0.03678	8.45905
SMRMF	0.00027	0.00268	0.00112	0.00024	554	0.00952	6.75383
SMRMF Tag	0.00021	0.00112	0.00022	0.00015	652	0.01040	6.72105
SMRMF LOD	0.00011	0.00071	0.00022	0.00010	1234	0.02770	8.17090
WRMF	0.00295	0.02086	0.01151	0.00316	257	0.01195	7.28991
WRMF Tag	0.00252	0.01791	0.00898	0.00246	452	0.01346	7.45752
WRMF LOD	0.00209	0.01518	0.00721	0.00212	552	0.01376	7.48693
MostPopular	0.00005	0.00018	0.00000	0.00004	14	0.00097	3.48598

Table 5.16: Comparative results on `Last.fm` dataset, Top-10 recommendation list and relevance threshold of 7/10.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-`fsr` LOD	0.03355	0.03254	0.04071	0.03832	899	0.02222	7.88347
CB-`fsr` LOD	0.02304	0.02246	0.02695	0.02458	772	0.01347	6.77499
Jaccard-`fsr` Tag	<u>0.04812</u>	<u>0.04328</u>	<u>0.04865</u>	<u>0.05145</u>	<u>1058</u>	<u>0.03078</u>	<u>8.45781</u>
CB-`fsr` Tag	**0.04930**	**0.04582**	**0.06167**	**0.05507**	1008	0.02621	8.07217
Hybrid-`fsr` 10004	0.03376	0.03236	0.04133	0.03912	946	0.02401	8.01508
Hybrid-`fsr` 5002	0.02583	0.02294	0.02731	0.02841	1028	**0.03232**	**8.57696**
Hybrid-`fsr` 2051	0.00911	0.00762	0.00796	0.01011	907	0.02905	8.44240
Hybrid-`fsr` 1250	0.00493	0.00384	0.00470	0.00568	656	0.02387	8.22125
BPRMF	0.00407	0.00553	0.00372	0.00370	253	0.00599	6.28848
BPRMF Tag	0.00000	0.00000	0.00000	0.00000	151	0.00099	3.43110
BPRMF LOD	0.00107	0.00091	0.00057	0.00091	**1112**	0.02934	8.14561
SMRMF	0.00482	0.00403	0.00351	0.00512	364	0.00647	6.24558
SMRMF Tag	0.00182	0.00155	0.00074	0.00136	411	0.00648	5.99848
SMRMF LOD	0.00086	0.00085	0.00035	0.00064	777	0.01832	7.51538
WRMF	0.01983	0.02150	0.02291	0.01936	186	0.00839	6.79522
WRMF Tag	0.01426	0.01281	0.01567	0.01406	300	0.00820	6.71812
WRMF LOD	0.01050	0.01013	0.00997	0.00917	334	0.00829	6.75719
MostPopular	0.00011	0.00013	0.00002	0.00007	9	0.00043	2.50907

Table 5.17: Comparative results on `Last.fm` dataset, Top-5 recommendation list and no relevance threshold.

Alg	P@N	R@N	nDCG@N	EPC	ADiv@N	Gini	SE
Jaccard-fsr LOD	0.02556	0.04863	0.04784	0.03099	1254	0.02848	8.24498
CB-fsr LOD	0.01929	0.03838	0.03477	0.02138	1196	0.01932	7.34914
Jaccard-fsr Tag	0.03735	0.06844	0.05942	0.04263	1544	**0.03931**	8.80789
CB-fsr Tag	**0.03794**	**0.07064**	**0.07165**	**0.04510**	1519	0.03539	8.52017
Hybrid-fsr 10004	0.02653	0.05034	0.04874	0.03214	1325	0.03072	8.38414
Hybrid-fsr 5002	0.01902	0.03343	0.03220	0.02273	1421	0.03900	**8.85578**
Hybrid-fsr 2051	0.00702	0.01214	0.01071	0.00832	1191	0.03486	8.71436
Hybrid-fsr 1250	0.00370	0.00617	0.00600	0.00457	807	0.02910	8.54335
BPRMF	0.00348	0.00902	0.00495	0.00338	334	0.00768	6.66212
BPRMF Tag	0.00032	0.00073	0.00028	0.00021	242	0.00226	4.75911
BPRMF LOD	0.00096	0.00188	0.00084	0.00088	**1591**	0.03678	8.45905
SMRMF	0.00407	0.00723	0.00471	0.00447	554	0.00952	6.75383
SMRMF Tag	0.00225	0.00402	0.00155	0.00181	652	0.01040	6.72105
SMRMF LOD	0.00080	0.00156	0.00058	0.00067	1234	0.02770	8.17090
WRMF	0.01656	0.03566	0.02847	0.01711	257	0.01195	7.28991
WRMF Tag	0.01372	0.02467	0.02217	0.01369	452	0.01346	7.45752
WRMF LOD	0.01099	0.02072	0.01522	0.00991	552	0.01376	7.48693
MostPopular	0.00027	0.00089	0.00021	0.00019	14	0.00097	3.48598

Table 5.18: Comparative results on Last.fm dataset, Top-10 recommendation list and no relevance threshold.

Results Discussion

FF shows interesting performance in terms of accuracy, diversity, and novelty on all the experimental settings and it results as a highly competitive approach compared to other pure-collaborative and hybrid variants of state-of-the-art algorithms. In particular, we see that on the MovieLens,LibraryThing, and Last.fm datasets, at least a variant of FF gets the best results for the evaluated accuracy metrics. On the other hand, our Feature Factorization is not the absolute champion when compared on the MovieLens, and Last.fm datasets considering diversity metrics. However, if we consider the trade-off between accuracy, novelty, and diversity, we see that FF is the best performing algorithm. The differences between the different behaviors on the three datasets have been detailed in the previous discussion, and summing up they can be explained by looking at different dimensions of the datasets (LOD, Tag side information; quality of descriptions, sparsity of the datasets, and popularity bias). In Last.fm, we have rescaled the users' feedback represented as the number of times they played a song and normalized it on a 1-10 scale. This could have affected the final results especially in terms of accuracy for all the algorithms. If we consider the feature-augmented dataset, by looking at the data represented in Table 5.4 the first observation we make is that the number of features in `Last.fm` is two orders of magnitude higher than the number of items while in `LibraryThing` it is just one. Then, the decrease in performance of `FF` may be also attributed to the curse of dimensionality problem. Moreover, a deeper investigation of the quality of the adopted `LOD` dataset is needed. A few papers have been published on this topic [291, 58], however, still there is not a community-endorsed metric to evaluate the quality of the knowledge encoded in a Linked Data dataset for recommendation tasks.

5.5 Conclusion

In this line of research, we have introduced `FF`, a novel algorithm that bases on feature recommendation as an intermediate step for computing *top-N* items recommendation lists. The main idea behind `FF` is that feature relevance in a user profile plays a key role in the selection and rating of an item in a collection. Based on this observation we have developed an algorithm that shifts the recommendation problem from a user-item space to a user-feature one. In this new space, we have introduced the notion of feature relevance and feature rating. We have combined them with well-known factorization techniques computing rating and relevance for each feature unknown to the user. Then, by combining the values associated with the features composing an item we have predicted a *top-N* recommendation list of items. We have compared `FF` with well-known factorization techniques (both pure collaborative and hybrid variants with side information) on three datasets in the domains of movie, books, and music. In all the datasets `FF` results as the best algorithm in terms of a trade-off between accuracy, diversity, and novelty of results. This can be considered as a strong clue to confirm our intuition that recommending items via feature ranking is a feasible way to develop content-aware recommendation engines. As future work, we are investigating the behavior of `FF` with different factorization techniques in the item-feature space. Moreover, since we have collected content-based data from Linked Open Data datasets, an analysis of the influence of such datasets on the recommendation results is also in progress. Another aspect we are willing to deepen is related to results explanation. Indeed, very interestingly, item recommendation via feature ranking paves the way to new proposals for explanation services.

Chapter 6

Metadata to address Cold-start problem

6.1 Introduction

Cross-domain recommendation has recently emerged as a potential solution to the cold start problem in recommender systems [55], aiming to mitigate the lack of data by exploiting user preferences and item attributes in domains distinct but related to the target domain. In this line, most of the cross-domain approaches proposed so far are based on collaborative filtering [71], exploiting user preferences as a bridge to relate source and target domains, and ignoring the content of the items. Hence, they benefit from the fact that they do not need to perform any kind of analysis of item contents, which are in general highly heterogeneous across domains, and whose inter-domain relationships may be difficult to be establish.

These difficulties, however, can be addressed nowadays thanks to the Semantic Web initiative [241], and more specifically to its reference implementation the Linked Open Data (LOD) project [39], which has originated a large number of inter-linked knowledge repositories publicly available in the Web, following the Semantic Web standards for data representation and access. Hence, in the current Web there is a wide array of structured data sources with information of items belonging to a variety of domains, such as history, arts, science, industry, media and sports, to name a few. This information not only consists of particular multimedia contents and associated metadata, but also explicit, semantic relations between items and metadata.

Motivated by the availability of large amounts of item metadata and semantic relations in the Linked Data cloud, we aim to address the cross-domain recommendation problem not only focusing on user preferences and item attributes, but also exploiting content-based relations between items from different domains. More specifically, we propose to use the set of `LOD` semantic features and relations as inter-domain links for supporting knowledge transfer across domains, enabling cross-domain item similarities, and providing recommendations for cold start users in the target domain.

Previous work has proposed graph-based algorithms to address the recommendation problem in heterogeneous datasets [288, 194], analyzing the topology of semantic networks to jointly exploit user preferences and item metadata. These approaches have been shown to be effective for recommendation, but suffer from computational issues caused by the size of the semantic networks, which are in general very large. Differently, we avoid these issues by working in two steps. First, we exploit the semantic networks to compute inter-domain similarities that link items from different domains. Then, we leverage the computed similarities in hybrid Matrix Factorization (MF) models for recommendation, which no longer need to deal with the whole networks.

Therefore, this research line has led to the development of novel, effective hybrid matrix factorization models that jointly exploit user preferences and item metadata for cross-domain recommendation. Moreover, we adapt a fast learning algorithm by [213] for efficiently building our models, and evaluate them in cold start scenarios on several domains, in terms of both precision and diversity.

We evaluate the performance of the proposed models using a dataset of Facebook[1] *likes* about books, movies and music. In order to obtain semantic metadata for the different items, we first mapped the items in our dataset to entities in `LOD` by means of `SPARQL` queries, and then extracted their attributes and relations to enhance the item profiles.

In a first experiment, we compared several state-of-the-art semantic similarity metrics for content-based recommendation, aiming to understand which is more suitable for later injecting in our cross-domain MF models, and achieved the best results using the link-based approach by [175]. Second, we evaluated the ranking precision and diversity of the recommendations computed by the proposed models. We show that, depending on the involved source and target domains, our models generate more accurate suggestions than the baselines in severe cold start situations. Moreover, the proposed approaches provide a better trade-off between accuracy and diversity, which are in general difficult to balance.

We point out that the presented approaches can be effectively used if the underlying LOD knowledge graph encodes direct or indirect connections between items in different domains. In fact, we need to compute semantic similarity values between items not belonging to the same domain. These connections are quite common for knowledge domains with some degree of information overlapping such as in the case of music, movies, and books but, in case they are missing or rare, this may result as a limitation for the performances of the approaches we introduce here.

The reminder of the chapter is structured as follows. In 6.2, we revise related work on cross-domain recommender systems, focusing on those approaches that are based on Matrix Factorization. In 6.3, we present the developed cross-domain hybrid matrix factorization models. Next, in 6.4, we report and analyze the empirical results achieved in the experiments conducted to analyze user cold start situations. Finally, in 6.5 we end with some conclusions and future research lines.

[1] Facebook online social networking, https://www.facebook.com

6.2 Related work

In this section, we survey the state of the art on cross-domain recommender systems. First, in 6.2.1 we describe the cross-domain recommendation problem and present a categorization of the approaches, giving representative examples of each category. Next, in 6.2.2 we focus on those cross-domain recommendation approaches that use the matrix factorization technique to bridge the source and target domains.

6.2.1 Cross-domain recommender systems

Nowadays, the majority of recommender systems offer recommendations for items belonging to a single domain. For instance, Netflix[2] recommends movies and TV shows, Spotify[3] recommends songs and music albums, and Barnes & Noble[4] recommends books. These domain-specific systems have been successfully deployed by numerous web platforms, and the single-domain recommendation functionality is not perceived as a limitation, but rather pitched as a focus on a certain market.

Nonetheless, in large e-commerce sites such as Amazon.com[5] and eBay[6] users often provide feedback for items from multiple domains, and in social networks like Facebook[7] and Twitter[8] users express their tastes and interests for a variety of topics. It may, therefore, be beneficial to leverage all the available user data provided in various systems and domains, in order to generate more encompassing user models and better recommendations. Instead of treating each domain (e.g., movies, music and books) independently, knowledge acquired in a *source* domain could be transferred to and exploited in another *target* domain. The research challenge of transferring knowledge, and the business potential of delivering recommendations spanning across multiple domains, have triggered an increasing interest in *cross-domain recommendations*.

[2] Netflix streaming media and video provider, `https://www.netflix.com`

[3] Spotify digital music service, `https://www.spotify.com`

[4] Barnes & Noble online bookseller, `http://www.barnesandnoble.com`

[5] Amazon electronic commerce site, `https://www.amazon.com`

[6] eBay consumer-to-consumer and business-to-consumer sales, `http://www.ebay.com`

[7] Facebook social network, `https://www.facebook.com`

[8] Twitter online news and social networking service, `https://twitter.com`

The cross-domain recommendation problem has been addressed from various perspectives in different research areas. It has been handled by means of user preference aggregation and mediation strategies for cross-system personalization in User Modeling [4, 36, 242], as a potential solution to mitigate the cold start and sparsity problems in Recommender Systems [71, 246, 263], and as a practical application of knowledge transfer in Machine Learning [105, 156, 202]. Focusing on how knowledge is exploited by cross-domain recommender systems, in [55] we categorized existing works according a two-level taxonomy.

- *Aggregating knowledge*. Knowledge from various source domains is aggregated to perform recommendations in a target domain. Depending on the stage in the recommendation process where the aggregation is performed we can further distinguish three cases. First, we find approaches that *merge user preferences* e.g., ratings, tags, transaction logs, and click-through data. The aggregation can be done by means of a multi-domain rating matrix [35, 229], using a common representation for user preferences such as social tags [257, 4, 94] or semantic concepts [140], linking the preferences via a multi-domain graph [71, 263], or mapping user preferences to domain-independent features such as personality traits [54] or user-item interaction features [171]. In the second case, user modeling data from various recommender systems is *mediated* to improve target recommendations. For instance, [35, 264, 242] import user neighborhoods and user-user similarities computed in the source domain into the target. Finally, some approaches directly *combine* single-domain recommendations, e.g., rating estimations [35, 106] and rating probability distributions [303].

- *Linking and transferring knowledge*. Knowledge linkage or transfer between domains is established to support recommendations. In this case, we find methods that (i) *link domains* by a common knowledge such as item attributes [65], association rules [54], semantic networks [93, 140], and inter-domain correlations [295, 246, 230]; methods that (ii) *share latent features* between source and target domains factor models, either by using same model parameters [201, 128, 119] in both factorizations, or by introducing new parameters that extend the factorizations [89, 92]; and methods that (iii) *transfer rating patterns* extracted by co-clustering the source domain rating matrix and exploit them in the target domain [157, 105, 70]. After defining the problem, in [200] three different knowledge transfer strategies for collaborative recommendation with auxiliary data (TL-CRAD) are introduced: (i) adaptive knowledge transfer, (ii) collective knowledge transfer and (iii) integrative knowledge transfer. Then, for each of them the author surveys related work with reference to different knowledge strategies with an emphasis on: transfer via prediction rule, transfer via regularization and transfer via constraint.

In terms of the goals addressed by cross-domain recommenders, we find great diversity among the reviewed approaches. Most proposals focus on improving accuracy by reducing data sparsity [156, 246, 56, 295, 202, 263, 171, 302]. In many domains, the average number of ratings per user and item is low, which may negatively affect the quality of the recommendations. Data collected outside the target domain can increase the rating density, and thus may upgrade the recommendation quality. Others seek to enhance user models, which may have personalization-oriented benefits such as (i) discovering new user preferences for the target domain [254, 258], (ii) enhancing similarities between users and items [3, 36], and (iii) measuring vulnerability in social networks [107, 132]. Cross-domain methods have also been applied to bootstrap recommender systems by importing preferences from another source outside the target domain [242], and have been proposed to improve the diversity of recommendations by providing better coverage of the range of user preferences [281]. Finally, a few approaches have dealt with the new user problem [128, 229, 264, 89]. When a user starts using a recommender system, this has no knowledge of the user's tastes and interests, and cannot produce personalized recommendations. This may be solved by exploiting the user's preferences collected in a different source domain.

We observe that addressing the cold-start has been barely investigated as in [166] where the authors present a neighborhood-based algorithm for the dual cold-start problem. The generalization of of items and users into a cluster level to obtain high-quality relations also in cold start scenario is the focus of [176]. They first employ biased matrix factorization to map rating matrix int lower-dimension latent spaces. After this step they apply the K-means clustering algorithm to categorize users and items. Cold start is also the main topic of [283] where the authors propose a novel approach to cross-system personalization based on two assumptions: the existence of a user model that could be shared among platforms and that a specific system can maintain (and provide) the user models built by its system.

As we shall present in 6.3, we aim to deal with the cold-start problem by means of novel matrix factorization models that jointly exploit user ratings and item metadata. Before, in 6.2.2, we revise state of the art cross-domain recommender systems based on matrix factorization.

6.2.2 Matrix factorization-based cross-domain recommender systems

Although matrix factorization models can be applied in cross-domain approaches based on knowledge aggregation –essentially as a standard recommendation problem once the user preferences from both domains are combined– they have been mostly used in knowledge linkage or transfer approaches. In these settings, latent factors from source and target domains are either shared or related in order to establish the bridge between the domains.

One way of linking domains explored in previous works exploits inter-domain similarities by integrating them into the probabilistic matrix factorization method [232]. Specifically, such similarities are imposed as constraints over user or item latent factors when jointly factorizing rating matrices. For instance, [56] proposed an approach in which inter-domain similarities are implicitly learned from data, as model parameters in a non-parametric Bayesian framework. Since user feedback is used to estimate the similarities, user overlap between the domains is required. Addressing the sparsity problem, [295] adapted the probabilistic matrix factorization method to include a probability distribution of user latent factors that encodes inter-domain correlations. One strength of this approach is that user latent factors shared across domains are not needed, allowing more flexibility in capturing the heterogeneity of domains. Instead of automatically learning implicit correlations in the data, [246] argued that explicit common information is more effective, and relied on shared social tags to compute cross-domain user-to-user and item-to-item similarities. Similarly to previous approaches, rating matrices from the source and target domains are jointly factorized; but in this case user and item latent factors from each domain are restricted, so that their product is consistent with the tag-based similarities.

Latent factors shared between domains can be exploited to support cross-domain recommendations. In this context, two types of approaches have been studied to perform the actual transfer of knowledge; namely, *adaptive* and *collective* models. In the former, latent factors are learned in the source domain, and are integrated into a recommendation model in the target domain, while in the latter, latent factors are learned simultaneously optimizing an objective function that involves both domains. [202] addressed the sparsity problem in the target domain following the adaptive approach, proposing to exploit user and item information from auxiliary domains where user feedback may be represented differently. In particular, they studied the case in which users express binary like/dislike preferences in the source domain, and utilize 1-5 ratings in the target domain. Their approach performs singular value decomposition (SVD) in each auxiliary domain, in order to separately compute user and item latent factors, which are then shared with the target domain. Specifically, transferred factors are integrated into the factorization of the rating matrix in the target domain and added as regularization terms so that specific characteristics of the target domain can be captured. Latent factors can also be shared in a collective way, as studied by [201]. In this case, instead of learning latent features from the source domains and transferring them to the target domain, the authors proposed to learn the latent features simultaneously in all the domains. Both user and item factors are assumed to generate the observed ratings in every domain, and, thus, their corresponding random variables are shared between the probabilistic factorization models of each rating matrix. Moreover, the factorization method is further extended by incorporating another set of factors that capture domain-dependent information, resulting in a tri-factorization scheme. A limitation of the proposed approach is that the users and items from the source and target domains have to be identical. Instead of focusing on sharing latent factors, [89], and [92] studied the influence of social tags on rating prediction, as a knowledge transfer approach for cross-domain recommendations. The authors presented a number of models based on the SVD++ algorithm [148] to incorporate the effect of tag assignments into rating estimation. The underlying hypothesis is that information about item annotation in a source domain can be exploited to improve rating prediction in a target domain, as long as a set of common tags between the domains exists. In the proposed models, tag factors are added to the latent item vectors, and are combined with user latent features to compute rating estimations. The difference between these models is in the set of tags considered for rating prediction. In all the models knowledge transfer is performed through the shared tag factors in a collective way, since these are computed jointly for the source and the target domains. [128] presented a more complex approach that takes domain factors into account. There, the authors argue that user-item dyadic data cannot fully capture the heterogeneity of items, and that modeling domain-specific information is essential to make accurate predictions in a setting where users typically express their preferences in a single domain. They referred to this problem as the *unacquainted world*, and proposed a tensor factorization algorithm to exploit the triadic user-item-domain data. In that method, rating matrices from several domains are simultaneously decomposed into shared user, item, and domain latent factors, and a genetic algorithm automatically estimates optimal weights of the domains. In a recent work, [302], the authors propose a two-step approach where the latent factors learned via MF for both the source and target domains are linked by training a deep neural network (DNN) representing their connections. Interestingly, the training process of the DNN is driven by the sparsity degrees of individual users and items in the source and target domains. Contextual and content-based information is exploited in [260] to cluster users in the source domain prior to a tensor factorization. The proposed Cross Domain- Multi Dimensional Tensor Factorization (CD-MDTF) mitigates the sparsity and cold-start problem by transferring the aggregated knowledge from the source domain to target domain. An approach based on linking and transferring knowledge is proposed in [299] where the main assumption is that correspondences among entities is different domains are unknown but can be computed with a cost. Starting from this assumption, the authors propose a unified framework aimed at actively mapping entities in different domain and then transferring knowledge via collaborative filtering. This latter step leverages partial mappings among entities for knowledge transfer. The authors also show how to integrate in their framework various extended matrix factorization techniques in a transfer learning manner. An emphasis on the meaningfulness of the knowledge extracted from the source domain to the target domain is the main topic if [293]. A clustering step among users and items is performed both in the source and target before a matrix factorization. Then, by comparing the resulting matrices, it is possible to evaluate the consistency of the information transfer.

Rather than sharing user or item latent factors for knowledge transfer, a different set of approaches analyzes the structure of rating data at the community level. These methods are based on the hypothesis that even when their users and items are different, close domains are likely to have user preferences sampled with the same population. Therefore, latent correlations may exist between preferences of groups of users for groups of items, which are referred to as *rating patterns*. In this context, rating patterns can act as a bridge that relates the domains, such that knowledge transfer can be performed in either adaptive or collective manners. In the adaptive setting, rating patterns are extracted from a dense source domain [156, 105]. In the collective setting, data from all the domains are pulled together and jointly exploited, even though users and items do not overlap across domains [157]. In [119], the authors propose to alleviate the data sparsity problem in a target domain by transferring rating patterns from multiple incomplete source domains. The proposed approach extracts rating patterns from a sparse source domain that are eventually combined with collaborative filtering to approximate the target domain and predict missing values. In particular, they take into account the effects related to negative transfer to obtain a more robust recommendation.

6.3 Matrix factorization models for cross-domain recommendation

In this section we refer to state-of-art models and optimization techniques explained in Sections: 3.2.3, 3.2.3, 3.2.5, 3.2.4, 3.3.1. In contrast to previous works that rely on graph-based methods for exploiting semantic metadata, the proposed approach first computes inter-domain content-based similarities between the items, and then exploits these similarities to regularize the joint learning of matrix factorizations in the source and target domains. In particular, we present three alternative hybrid models that make different assumptions about the relationships between source and target domain item latent factors, simultaneously exploiting user preferences and item metadata.

Moreover, in the chapter we detail our adaptations of the fast alternating least squares training algorithm for matrix factorization proposed by [213], in order to deal with the increased complexity of our models, which not only learn the auxiliary source domain user preferences, but also the item metadata used to bridge the domains.

Items from different domains tend to have very diverse attributes that are not straightforward related. For instance, a book may be characterized by its *author* or by its *book genres*, and a movie can be described using its *cast, director* or *movie genres*. In fact, content-based features are often different between domains, and even when they refer to related concepts, such as *book genres* and *movie genres*, the features may not be directly aligned, e.g., *funny movies* vs. *comedy books*.

In order to overcome the heterogeneity of features of items from different domains, we propose to exploit Linked Data for linking entities from multiple and diverse domains. Specifically, we map the items in our datasets to entities in DBpedia, a multi-domain repository that provides a semantic-based, structured representation of knowledge in Wikipedia. In 6.4.1 we shall describe the process of mapping items to semantic entities from DBpedia. Once the items are mapped to their corresponding entities, we use the DBpedia graph to compute semantic similarities between such entities, mining both the attributes and the structure of the graph with semantic relations. More specifically, we exploit the information in DBpedia to compute a semantic similarity matrix $\mathbf{S} \in \mathbb{R}^{|\mathscr{I}_S| \times |\mathscr{I}_T|}$ between the source domain items $\mathscr{I}_S$ and the target domain items $\mathscr{I}_T$:

$$s_{ij} = sim(i, j), \qquad i \in \mathscr{I}_S, j \in \mathscr{I}_T \tag{6.1}$$

In 6.4.4 we shall report recommendation performance results by using several semantic similarity metrics from the state of the art.

The computed inter-domain item similarities are then used to *link* the domains for cross-domain recommendation. In the cold start, when a user has rated a few (if any) items in the target domain, a recommender system could suggest the user with items in the target domain that are semantically similar to those the user liked in the source domain. Hence, the system could be effective only if there is an overlap of users between the domains. Moreover, even cold start users in the target domain should have some preferences in the source domain.

In the next subsections we present our three recommendation models based on the exploitation of semantic similarities to regularize item factors in MF, so that similar items from different domains tend to have similar parameters. In this way, even if the user's preferences in the target domain are unknown, a recommender system could suggest the user with target items that are most similar to those she preferred in the source.

6.3.1 Regularization through similarity prediction

The first semantic-based matrix factorization cross-domain model we propose is based on the assumption that latent vectors of related items should explain the items semantic similarities, in addition to the users' preferences. That is, we not only seek to predict the preferences $r_{ui} \approx \langle \vec{p}_u, \vec{q}_i \rangle$, but also the inter-domain similarities $s_{ij} \approx \langle \vec{q}_i, \vec{q}_j \rangle$, where $i \in \mathscr{I}_S$ and $j \in \mathscr{I}_T$.

Hence, our model jointly factorizes the rating and inter-domain item similarity matrices that link the source and target domains. Let $\mathscr{U} = \mathscr{U}_S \cup \mathscr{U}_T$ be the set of all users, which we assume overlaps between the domains, and let $\mathscr{I} = \mathscr{I}_S \cup \mathscr{I}_T$ be the set of all items, which we assume do not overlap. Our model learns a latent vector

$\vec{p}_u \in \mathbb{R}^k$ for each user $u \in \mathscr{U}$, but separately models source and target domain items $\vec{q}_i$ and $\vec{q}_j$, with $i \in \mathscr{I}_S$ and $j \in \mathscr{I}_T$, as follows:

$$\mathscr{L}(\mathbf{P}, \mathbf{Q}_S, \mathbf{Q}_T) = \sum_{u \in \mathscr{U}} \sum_{a \in \mathscr{I}} c_{ua} \left(r_{ua} - \langle \vec{p}_u, \vec{q}_a \rangle\right)^2 + \lambda_C \sum_{i \in \mathscr{I}_S} \sum_{j \in \mathscr{I}_T} \left(s_{ij} - \langle \vec{q}_i, \vec{q}_j \rangle\right)^2 + \lambda \left(\|\mathbf{P}\|^2 + \|\mathbf{Q}_S\|^2 + \|\mathbf{Q}_T\|^2\right) \quad (6.2)$$

where $\mathbf{Q}_S$ and $\mathbf{Q}_T$ are matrices containing the item latent vectors as rows from the source and target domains, respectively. We note that the summation in the first term iterates over all items $a \in \mathscr{I}$ from both domains, as we want to factorize the source and target user-item preference matrices simultaneously. The cross-domain regularization parameter $\lambda_C > 0$ controls the contribution of the inter-domain semantic similarities; large values of the parameter will force items to have too similar latent vectors, whereas low values will result in limited transfer of knowledge between domains.

As in standard matrix factorization, we train our model using Alternating Least Squares. First, we fix $\mathbf{Q}_S$ and $\mathbf{Q}_T$, and solve analytically for each $\vec{p}_u$ by setting the gradient to zero. Since the user factors do not appear in the additional cross-domain regularization term, we obtain the same solution as for the baseline MF model (see 3.8):

$$\vec{p}_u = \left(\mathbf{Q}^\top \mathbf{C}^u \mathbf{Q} + \lambda \mathbf{I}\right)^{-1} \mathbf{Q}^\top \mathbf{C}^u \vec{r}_u \quad (6.3)$$

In order to simplify the notation, we have defined the matrix $\mathbf{Q}$ as the row-wise concatenation of $\mathbf{Q}_S$ and $\mathbf{Q}_T$. The matrix $\mathbf{C}^u$ is a diagonal matrix with the confidence values c_{ua} for all $a \in \mathscr{I}$, and the vector $\vec{r}_u$ contains the preferences of user u, again for all items $a \in \mathscr{I}$.

Next, we fix the user factors $\mathbf{P}$ and the target domain item factors $\mathbf{Q}_T$, and compute the optimal values for the source domain item factors. Again, by setting the corresponding gradient to zero and solving analytically we obtain:

$$\vec{q}_i = \left(\mathbf{P}^\top \mathbf{C}^i \mathbf{P} + \lambda_C \mathbf{Q}_T^\top \mathbf{Q}_T + \lambda \mathbf{I}\right)^{-1} \left(\mathbf{P}^\top \mathbf{C}^i \vec{r}_i + \lambda_C \mathbf{Q}_T^\top \vec{s}_i\right) \quad (6.4)$$

As previously, the vector $\vec{r}_i$ contains the preferences assigned to item i, and $\vec{s}_i$ is the i-th row of the inter-domain semantic similarity matrix $\mathbf{S}$. Finally, we proceed as before fixing $\mathbf{P}$ and $\mathbf{Q}_S$ to compute the optimal solution for the target domain item latent vectors:

$$\vec{q}_j = \left(\mathbf{P}^\top \mathbf{C}^j \mathbf{P} + \lambda_C \mathbf{Q}_S^\top \mathbf{Q}_S + \lambda \mathbf{I}\right)^{-1} \left(\mathbf{P}^\top \mathbf{C}^j \vec{r}_j + \lambda_C \mathbf{Q}_S^\top \vec{s}_j\right) \tag{6.5}$$

The computation of the optimal factors can be parallelized within each step, but the larger number of items to consider and the extra step required for the source domain greatly increase the training time with respect to the MF baseline. In order to address this issue, we adapt the fast RR1 training algorithm for ALS proposed by [213]. Since the computation of the user factors is the same as in the original MF model, the procedure remains the same for the P-step. For the source domain Q-step, by inspecting 6.2 and 6.4, we note that the additional terms that arise from the inter-domain similarities can be treated just like user preferences as follows. For each source item i:

1. Generate examples for each rating r_{ui} as for baseline MF (see [213])
2. For each target item $j \in \mathscr{I}_T$:
    - Generate an input example $\vec{x}_j := \vec{q}_j$.
    - Use the similarity as the dependent variable, $y_j := s_{ij}$.
    - Use a constant confidence value $c_j := \lambda_C$.
    - The parameter to optimize is $\vec{w} := \vec{q}_j$.

The above procedure will produce the similarity terms of 6.4, which can be defined by means of the confidence matrix $\tilde{\mathbf{C}}^i = \lambda_C \mathbf{I}$. The procedure for the target domain Q-step is completely analogous.

6.3.2 Regularization based on item neighborhoods

Our second semantic-based matrix factorization cross-domain model exploits the item semantic similarities in a different fashion. Instead of forcing pairwise item interactions to reproduce the observed similarity values, the approach we present here leverages $\mathbf{S}$ to regularize the item latent vectors, so that feature vectors of similar items are pushed together in the latent space. Intuitively, items that are semantically similar should also have similar latent parameters.

As previously, let $\mathscr{U} = \mathscr{U}_S \cup \mathscr{U}_T$ and $\mathscr{I} = \mathscr{I}_S \cup \mathscr{I}_T$ be the sets of all users and items, respectively. Our approach jointly factorizes the source and target domain rating matrices, and regularizes similar item factors proportionally to the items similarity. However, instead of considering all the potentially similar source domain items, we limit the regularization of a target domain item $j \in \mathscr{I}_T$ to its neighborhood, i.e., to the set $N(j) \subseteq \mathscr{I}_S$ of the top-n most similar source domain items:

$$\begin{aligned} \mathscr{L}(\mathbf{P}, \mathbf{Q}_S, \mathbf{Q}_T) = & \sum_{u \in \mathscr{U}} \sum_{a \in \mathscr{I}} c_{ua} \left(r_{ua} - \langle \vec{p}_u, \vec{q}_a \rangle \right)^2 \\ & + \lambda_C \sum_{j \in \mathscr{I}_T} \sum_{i \in N(j)} s_{ij} \left\| \vec{q}_j - \vec{q}_i \right\|^2 + \lambda \left(\sum_{u \in \mathscr{U}} \|\vec{p}_u\|^2 + \sum_{a \in \mathscr{I}} \|\vec{q}_a\|^2 \right) \end{aligned} \quad (6.6)$$

We note that items with greater similarity values are more heavily regularized, whereas items with values of $s_{ij} \approx 0$ in their neighborhoods are barely affected. However, it may still be convenient to regularize such items so that they benefit from cross-domain information, and thus may be eligible for recommendation to cold start users. Therefore, we also experiment normalizing the similarity scores in the item neighborhoods so that $\sum_{i \in N(j)} s_{ij} = 1$. In this way all target items are equally regularized, but each is affected by its source domain neighbors proportionally to their similarity scores.

By assigning latent vectors to target domain items close to those of similar source domain items, our model is able to generate recommendations in cold start settings. Specifically, let $\vec{q}_j$ be the latent vector learned for target item $j \in \mathscr{I}_T$, and let $\vec{q}_i$ be the latent vector of source item $i \in \mathscr{I}_S$, which we assume is semantically similar to j. Our model will regularize both factors so that their distance $\left\| \vec{q}_j - \vec{q}_i \right\|$ is small, or equivalently, $\vec{q}_j \approx \vec{q}_i$. Consider now a cold start user u who only provided preferences in the source domain, so that her corresponding latent vector $\vec{p}_u$ is therefore only adjusted using source domain preferences. In standard MF, it is not guaranteed that $\vec{p}_u$ will extrapolate to the target domain, and will provide an accurate prediction for $\vec{q}_j$. In contrast, our model ensures that $\langle \vec{p}_u, \vec{q}_j \rangle \approx \langle \vec{p}_u, \vec{q}_i \rangle$, i.e., target domain items yield relevance prediction scores close to that of similar source domain items. Hence, u will be recommended with a target domain item j if the user liked the source domain item i, or if i would be recommended to u in the source domain.

Once more, we train our neighborhood-based matrix factorization model using Alternating Least Squares. As in the previous model, the user factors are not affected by the extra regularization, and can be computed again using 6.3, leaving the P-step unchanged. For the target domain item factors $\vec{q}_j$ we proceed as usual, fixing the user and source item factors, and finding the values such that $\frac{\partial \mathscr{L}}{\partial \vec{q}_j} = 0$, which yields the solution:

$$\vec{q}_j = \left[\mathbf{P}^\top \mathbf{C}^j \mathbf{P} + \left(\lambda + \lambda_C \sum_{i \in N(j)} s_{ij} \right) \mathbf{I} \right]^{-1} \left(\mathbf{P}^\top \mathbf{C}^j \vec{r}_j + \lambda_C \sum_{i \in N(j)} s_{ij} \vec{q}_i \right) \tag{6.7}$$

Repeating the same procedure for the source item factors $\vec{q}_i$ we obtain:

$$\vec{q}_i = \left[\mathbf{P}^\top \mathbf{C}^i \mathbf{P} + \left(\lambda + \lambda_C \sum_{j \in N^{-1}(i)} s_{ij} \right) \mathbf{I} \right]^{-1} \left(\mathbf{P}^\top \mathbf{C}^i \vec{r}_i + \lambda_C \sum_{j \in N^{-1}(j)} s_{ij} \vec{q}_j \right) \tag{6.8}$$

where $N^{-1}(i)$ is the *inverse neighborhood* of item i, i.e., the set of target domain items that have i among their neighbors: $N^{-1}(i) = \{ j \in \mathscr{I}_T | i \in N(j) \}$.

Unlike the model presented in the previous section, we cannot apply RR1 directly by treating the new similarity terms as additional user preferences. Instead, we derive again the update rules for each component of the source and target domain item parameters. As mentioned before, user parameters remain unchanged. Let $j \in \mathscr{I}_T$ be a target item, and consider the optimization of the α-th component $q_{j\alpha}$ of its corresponding latent vector $\vec{q}_j$. We can rewrite the loss in 6.6 as a function only of $q_{j\alpha}$ as follows:

$$\begin{aligned} \mathscr{L}_\alpha(q_{j\alpha}) = \sum_{u \in \mathscr{U}} c_{uj} \left(e_{uj} - p_{u\alpha} q_{j\alpha} \right)^2 + \lambda q_{j\alpha}^2 \\ + \lambda_C \sum_{i \in N(j)} s_{ij} \left(q_{j\alpha} - q_{i\alpha} \right)^2 + \text{constant} \end{aligned} \tag{6.9}$$

where $e_{uj} \triangleq r_{uj} - \sum_{\beta \neq \alpha} p_{u\beta} q_{j\beta}$, and the constant includes terms that do not depend on $q_{j\alpha}$. If we set the derivative $\frac{\mathrm{d}\mathscr{L}_\alpha}{\mathrm{d}q_{j\alpha}} = 0$, we obtain:

$$q_{j\alpha} = \frac{\sum_{u\in\mathscr{U}} c_{uj} e_{uj} p_{u\alpha} + \lambda_C \sum_{i\in N(j)} s_{ij} q_{i\alpha}}{\sum_{u\in\mathscr{U}} c_{uj} p_{u\alpha}^2 + \lambda + \lambda_C \sum_{i\in N(j)} s_{ij}} \tag{6.10}$$

Using the optimizations described in [213], the computational cost of the above formula for all items is $\mathscr{O}(k^2|\mathscr{U}| + k|\mathscr{R}| + n|\mathscr{I}_T|)$, since all the neighborhoods are formed using the top n most similar items, $|N(j)| \leq n$. Applying the same procedure to the source domain item factor $\vec{q}_i$ we obtain:

$$q_{i\alpha} = \frac{\sum_{u\in\mathscr{U}} c_{ui} e_{ui} p_{u\alpha} + \lambda_C \sum_{j\in N^{-1}(i)} s_{ij} q_{j\alpha}}{\sum_{u\in\mathscr{U}} c_{ui} p_{u\alpha}^2 + \lambda + \lambda_C \sum_{j\in N^{-1}(i)} s_{ij}} \tag{6.11}$$

The main difference with respect to 6.10 is that the sets $N^{-1}(i)$ are not bounded, as a source item can potentially be the neighbor of an arbitrary number of target items, so that $|N^{-1}(i)| \leq |\mathscr{I}_T|$, resulting in a theoretical worst-case cost of $\mathscr{O}(k^2|\mathscr{U}| + k|\mathscr{R}| + |\mathscr{I}_S||\mathscr{I}_T|)$. We observe, however, that in practice most of the source items appear only in a few neighborhoods and that the algorithm is still very efficient.

6.3.3 Regularization based on item centroids

When neighbor source domain items are mutually diverse, the neighborhood-based model presented in the previous section may struggle to regularize a target domain item that has to be simultaneously close to all its neighbors. The model we propose in this section works like the neighborhood-based model, but, instead of using the neighbor source domain items *individually* in the regularization, it uses their *centroid* (average) latent vector:

$$\begin{aligned}\mathscr{L}(\mathbf{P}, \mathbf{Q}_S, \mathbf{Q}_T) = &\sum_{u\in\mathscr{U}} \sum_{a\in\mathscr{I}} c_{ua} (r_{ua} - \langle \vec{p}_u, \vec{q}_a \rangle)^2 \\ &+ \lambda_C \sum_{j\in\mathscr{I}_T} \left\| \vec{q}_j - \sum_{i\in N(j)} s_{ij} \vec{q}_i \right\|^2 + \lambda \left(\sum_{u\in\mathscr{U}} \|\vec{p}_u\|^2 + \sum_{a\in\mathscr{I}} \|\vec{q}_a\|^2 \right)\end{aligned} \tag{6.12}$$

The same considerations regarding the neighborhood $N(j)$ and the normalization of the similarity scores also apply to this model. However, the effect on the item relevance predictions for cold start users is different. Let $\vec{q}_j$ be an item in the target domain, and let $N(j)$ be its neighborhood of most similar source domain items. The regularization scheme in our centroid-based approach aims to minimize the distance $\left\|\vec{q}_j - \sum_{i\in N(j)} s_{ij}\vec{q}_i\right\|$, so that the latent vector of item j is close, *on average*, to those of the source items in $N(j)$, i.e., $\vec{q}_j \approx \sum_{i\in N(j)} s_{ij}\vec{q}_i$. Let u be a cold start user in the target domain that has some preferences in the source domain. Again, her feature vector $\vec{p}_u$ is only learned using the user's source preferences, and may not be reliable for computing relevance predictions for target domain items in standard MF. Our model, however, ensures that

$$\langle \vec{p}_u, \vec{q}_j \rangle \approx \left\langle \vec{p}_u, \sum_{i\in N(j)} s_{ij}\vec{q}_i \right\rangle = \sum_{i\in N(j)} s_{ij} \langle \vec{p}_u, \vec{q}_i \rangle$$

That is, the predicted relevance score is roughly the average of the relevance scores for the neighbor source domain items, weighted by their corresponding semantic similarity.

As in the previous models, the user parameters are not affected by the item regularization terms, and can be computed in the standard fashion using 6.3. For the target domain item factors $\vec{q}_j, j \in \mathscr{I}_T$, we set the gradient of 6.12 to zero to obtain:

$$\vec{q}_j = \left[\mathbf{P}^\top \mathbf{C}^j \mathbf{P} + (\lambda + \lambda_C)\mathbf{I}\right]^{-1} \left(\mathbf{P}^\top \mathbf{C}^j \vec{r}_j + \lambda_C \sum_{i\in N(j)} s_{ij}\vec{q}_i\right) \tag{6.13}$$

Comparing the above to 6.7 we observe that both are equivalent when $\sum_{i\in N(j)} s_{ij} = 1$, i.e., normalizing the similarity values has the same effect of than centroid-based regularization on the target domain item factors. The solution for source item factors $\vec{q}_i$, in contrast, has a different form:

$$\vec{q}_i = \left[\mathbf{P}^\top \mathbf{C}^i \mathbf{P} + \left(\lambda + \lambda_C \sum_{j\in N^{-1}(i)} s_{ij}^2\right)\mathbf{I}\right]^{-1} \cdot \left(\mathbf{P}^\top \mathbf{C}^i \vec{r}_i + \lambda_C \sum_{j\in N^{-1}(j)} s_{ij}\left(\vec{q}_j - \vec{z}_{j\setminus i}\right)\right) \tag{6.14}$$

where we have defined $\vec{z}_{j\setminus i} = \sum_{l\in N^{-1}(i), l\neq i} s_{lj}\vec{q}_l$ to simplify the notation. We note that, differently to the previous models, the computation of the source domain latent vectors cannot be parallelized, as the value of $\vec{q}_i, i \in \mathscr{I}_S$ depends on the values of other $\vec{q}_l, l \in \mathscr{I}_S$ through the parameter $\vec{z}_{j\setminus i}$. As a result, the training process can be slow when the set of source domain items is large. In our experiments, however, we observed that the time penalty of computing the source factors sequentially is usually compensated by the faster RR1 algorithm, although we do not provide any quantitative analysis as it falls out of the scope of this work.

In order to apply RR1 to our centroid-based approach, we derive again the solutions for each α-th coordinate separately. Once more, the solution for the user factors remains the same as it is not affected by the regularization terms. For the target domain item factors $\vec{q}_j$, we consider the loss in 6.12 as a function only of the α-th component $q_{j\alpha}$:

$$\begin{aligned} \mathscr{L}_\alpha(q_{j\alpha}) = &\sum_{u \in \mathscr{U}} c_{uj} \left(e_{uj} - p_{u\alpha} q_{j\alpha}\right)^2 + \lambda q_{j\alpha}^2 \\ &+ \lambda_C \left(q_{j\alpha} - \sum_{i \in N(j)} s_{ij} q_{i\alpha} \right)^2 + \text{constant} \end{aligned} \tag{6.15}$$

As previously, the constant includes terms that do not depend on $q_{j\alpha}$, and e_{uj} is defined as in 6.9. Setting the derivative $\frac{\mathrm{d}\mathscr{L}_\alpha}{\mathrm{d}q_{j\alpha}} = 0$ yields:

$$q_{j\alpha} = \frac{\sum_{u \in \mathscr{U}} c_{uj} e_{uj} p_{u\alpha} + \lambda_C \sum_{i \in N(j)} s_{ij} q_{i\alpha}}{\sum_{u \in \mathscr{U}} c_{uj} p_{u\alpha}^2 + \lambda + \lambda_C} \tag{6.16}$$

We note, once again, the similar form of the above solution with respect to the previous model in 6.10. If we apply the same procedure to the source domain item factors, we obtain:

$$q_{i\alpha} = \frac{\sum_{u \in \mathscr{U}} c_{ui} e_{ui} p_{u\alpha} + \lambda_C \sum_{j \in N^{-1}(i)} s_{ij} (q_{j\alpha} - \vec{z}_{(j\setminus i)\alpha})}{\sum_{u \in \mathscr{U}} c_{ui} p_{u\alpha}^2 + \lambda + \lambda_C \sum_{j \in N^{-1}(i)} s_{ij}^2} \tag{6.17}$$

The computational complexity for the target domain factors is equivalent to the model from the previous section, whereas for the source domain factors it is $\mathscr{O}(k^2|\mathscr{U}| + k|\mathscr{R}| + n|\mathscr{I}_S||\mathscr{I}_T|)$ in the worst case, which is similar to the neighborhood-based model since the size of the neighborhoods n is in general small.

6.4 Experiments

In a first experiment, we compared several state-of-the-art semantic similarity metrics for content-based recommendation, aiming to understand which is more suitable for later injecting in our cross-domain MF models, and achieved the best results using the link-based approach by [175]. Second, we evaluated the ranking precision and diversity of the recommendations computed by the proposed models. We show that, depending on the involved source and target domains, our models generate more accurate suggestions than the baselines in severe cold start situations. Moreover, the proposed approaches provide a better trade-off between accuracy and diversity, which are in general difficult to balance.

6.4.1 Dataset

Our dataset initially consisted of a large set of *likes* assigned by users to items in Facebook. Using the Facebook Graph API, a user's *like* is retrieved in the form of a 4-tuple with the following information: the identifier, name and category of the liked item, and the timestamp of the like creation, e.g., `{id: "35481394342", name: "The Godfather", category: "Movie", created_time: "2015-05-14T12:35:08+0000"}`. The name of an item is given by the user who created the Facebook page of such item. In this context, distinct names may exist for a particular item, e.g., *The Godfather, The Godfather: The Movie, The Godfather - Film series*, etc. Users thus may express likes for different Facebook pages which actually refer to the same item. Aiming to unify and consolidate the items extracted from Facebook likes, we developed a method that automatically maps the items names with the unique URIs of the corresponding DBpedia entities, e.g., `http://dbpedia.org/resource/The\_Godfather` for the identified names of *The Godfather* movie.

Linking items to DBpedia entities Given a particular item, we first identified DBpedia entities that are labeled with the name of the item. For such purpose, we launched a SPARQL query targeted on the subjects of triples that have `rdfs:label`[9] as property and the item title as object. The next query is an example for *The Matrix 2* title:

```
SELECT DISTINCT ?item WHERE {
  {
    ?item rdf:type dbo:Film .
    ?item rdfs:label ?name .
    FILTER regex(?name, "the.*matrix.*2", "i") .
  }
  UNION
  {
    ?item rdf:type dbo:Film .
    ?tmp dbo:wikiPageRedirects ?item .
    ?tmp rdfs:label ?name .
    FILTER regex(?name, "the.*matrix.*2", "i") .
  }
}
```

To resolve ambiguities in those names that correspond to multiple items belonging to different domains, we specify the type of the item we wanted to retrieve in each case. Specifically, the previous query includes a triple clause with `rdf:type`[10] (or `dbo:type`[11]) as property. Hence, in the given example, the subject `The Matrix 2` refers to the "movie" type, which is associated to the `dbo:Film` class in DBpedia. The item types were set from the item categories provided in Facebook, and their associated DBpedia and YAGO[12] classes[13] were identified by manual inspection of the `rdf:type` values of several entities. 6.1 shows the list of item types and DBpedia/YAGO classes we considered for the three domains of our dataset.

[9] Namespace for rdfs, `http://www.w3.org/2000/01/rdf-schema`
[10] Namespace for rdf, `http://www.w3.org/1999/02/22-rdf-syntax-ns\#`
[11] Namespace for dbo, `http://dbpedia.org/ontology`
[12] The YAGO knowledge base, `http://www.mpi-inf.mpg.de/yago-naga/yago`
[13] Namespace for yago, `http://dbpedia.org/class/yago`

Table 6.1: Considered item types and their DBpedia and YAGO classes for the three domains of the dataset.

	Item type	DBpedia/YAGO classes
Books	Book	dbo:Book, yago:Book102870092, yago:Book102870526
	Genre	yago:LiteraryGenres
	Writer	dbo:Writer, yago:Writer110794014
	Fictional character	dbo:FictionalCharacter, yago:FictionalCharacter109587565
Movies	Movie	dbo:Film, yago:Movie106613686
	Genre	dbo:MovieGenre, yago:FilmGenres
	Director	yago:FilmDirector110088200, yago:Director110014939
	Actor	dbo:Actor, yago:Actor109765278
	Fictional character	dbo:FictionalCharacter, yago:FictionalCharacter109587565
Music	Composition	dbo:Song, dbo:MusicalWork, dbo:Single, dbo:ClassicalMusicComposition, dbo:Opera
	Genre	dbo:MusicGenre, yago:MusicGenres, yago:MusicGenre107071942
	Album	dbo:Album, yago:Album106591815
	Musician	dbo:MusicalArtist, yago:Musician110339966, yago:Musician110340312, yago:Composer109947232
	Band	dbo:Band, yago:MusicalOrganization108246613

Moreover, running the previous query template we observed that a number of items were not linked to DBpedia entities because the labels corresponded to Wikipedia *redirection* webpages. In these cases, to reach the appropriate entities the query makes use of the `dbo:wikiPageRedirects` property. The result of the previous query for *The Matrix 2* is `http://dbpedia.org/resource/The\_Matrix\_Reloaded`, which actually is the DBpedia entity of the second movie in *The Matrix* saga. Here, it is important to note that thanks to the Wikipedia page redirect component we were able to link items whose names do not have a direct syntactic match with the label of its DBpedia entity, but with the label of a redirected entity, e.g., the *Matrix 2* title matches the `The Matrix Reloaded` entity.

Final semantically annotated dataset For every linked entity, we finally accessed DBpedia to retrieve the metadata that afterward will be used as input for the recommendation models. In this case, we launched a SPARQL query asking for all the properties and objects of the triples that have the target entity as subject. Following the example given before, such a query would be:

```
SELECT ?p ?o WHERE {
  dbr:The_Matrix_Reloaded ?p ?o .
}
```

This query returns all the DBpedia property-value pairs of the `dbr:The_Matrix_Reloaded`[14] entity. However, since our ultimate goal is item recommendation, we should only exploit metadata that may be relevant to relate common preferences of different users. Thus, we filtered the query results by considering certain properties in each domain. Specifically, 6.2 shows the list of DBpedia properties selected for each of the three domains of our dataset. Hence, for example, for the movie items, we would have as metadata the movies genres, directors, and actors, among others.

The items and relations shown in the table thus represent a *semantic network* that is automatically obtained from DBpedia for each particular domain. 6.3 shows

[14] Namespace for dbr, `http://dbpedia.org/resource`

statistics of the dataset for the three domains of interest, namely books, movies, and music. Additionally, users may express preferences in more than one domain. 6.4 shows the number of users shared between each pair of domains.

Semantically enriched item profiles Fixing *books*, *movies*, *musicians* and *bands* as the target items to be recommended, we can distinguish the following three types of item metadata obtained:

- *attributes*, which correspond to item-attribute entities associated to the considered item types of 6.2, and are distinct to the entities of target items, e.g., the genre(s), director(s) and actors of a particular movie.
- *related items*, which correspond to the item-item properties in 6.2 that derive related entities, e.g., the novel a movie is based on (`dbo:basedOn` property), the prequel/sequel of a movie (`dbo:previousWork` / `dbo:subsequentWork` properties), or the musicians belonging to a band (`dbo:bandMember` property).
- *extended attributes*, which correspond to attribute-attribute properties that generate extended item attributes, originally not appearing as metadata, e.g., the subgenres of a particular music genre (`dbo:musicSubgenre` property).

The above three types of item metadata constitute the semantically enriched item profiles that we propose to use in our recommendation models. We note that they differ from the commonly used content-based item profiles composed of plain attributes. We also note that in the conducted experiments, the results achieved by exploiting the enriched profiles were better than those achieved by only using item attributes.

6.4.2 Evaluation methodology and metrics

The evaluation of the proposed models was conducted utilizing a modified user-based 5-fold cross-validation strategy, based on the methodology by [146] for cold

Table 6.2: DBpedia properties considered as item metadata; *item* can be book, movie and composition, musician and band.

Relation	**DBpedia properties**
item – genre	`dct:subject, dbo:genre`
book – genre	`dbo:literaryGenre`
music genre – music genre	`dbo:musicSubgenre, dbo:musicFusionGenre, dbo:movement, dbo:derivative, dbo:stylisticOrigin`
item – author	`dbo:author, dbo:creator`
book – writer	`dbo:writer`
movie – actor, character, director	`dbo:starring, dbo:cinematograpy, dbo:director`
composition – musician	`dbo:artist, dbo:composer, dbo:musicComposer, dbo:musicalArtist, dbo:associatedMusicalArtist`
music item – album	`dbo:album`
band – musician	`dbo:bandMember, dbo:formerBandMember, dbo:musicalBand, dbo:associatedBand`
item – item, character	`dbo:series`
item – character	`dbo:portrayer`
item – item	`dbo:basedOn, dbo:previousWork, dbo:subsequentWork, dbo:notableWork`

Table 6.3: Statistics of the extracted dataset enriched with metadata.

	Books	**Movies**	**Music**
Users	1876	26 943	49 369
Items	3557	3901	5748
Likes	42 869	876 501	2 084 462
Sparsity (%)	99.4	99.2	99.3
Avg. items/user	22.85	32.53	42.22
Avg. users/item	12.05	224.69	362.64

Table 6.4: User overlap between domains. Right to each target, the ratio of shared users relative to the source domain.

	Target					
Source	**Books**	%	**Movies**	%	**Music**	%
Books	1876	100.0	1495	79.7	1519	81.0
Movies	1495	5.5	26 943	100.0	21 720	80.6
Music	1519	3.1	21 720	44.0	49 369	100.0

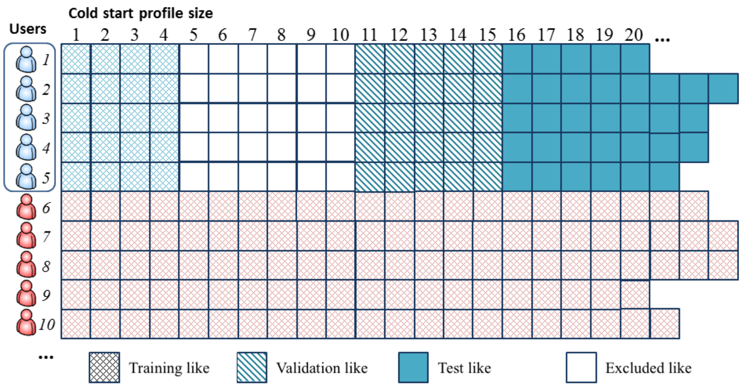


Figure 6.1: Overview of the cold start evaluation setting in a given cross-validation fold. The box indicates the test users in the current fold, whose profiles are split into training, validation, and testing sets. Different cold start profle sizes are simulated by sequentially adding *likes* to their training sets —four in the figure.

start evaluation. Our goal is to understand how the different approaches perform as the number of observed *likes* in the target domain increases. First, we divide the set of users into five subsets of roughly equal size. In each cross-validation stage, we keep all the data from four of the groups in the training set. Then, for each user *u* in the fifth group –the test users– we randomly split her *likes* into three subsets, as depicted in 6.1:

1. *Training data*, initially filled with *u*'s *likes* and iteratively downsampled discarding one by one to simulate different cold start profile sizes,

2. *Validation data* containing the set of *likes* used for tuning hyperparameters, and

3. *Testing data* used to compute the performance metrics.

The above procedure was modified for the cross-domain scenario by extending the training set with the full set of *likes* from the auxiliary domain, in order to obtain the actual training data for the predictive models. For each cold start profile size, we built the recommendation models using the data in the final training set. Then, for each test user, we generated a ranked list of the top 10 suggested items from the set of target domain items in the training set that are not yet known to the user. The performance is estimated from the output of each model and the test set using rank-based metrics. We note that in our evaluation, any item ranked after position 10 by the model is considered not relevant when computing the metrics, as we are interested in the more realistic setting where the user only examines a limited subset of the recommendations.

Regarding the metrics, we used the Mean Reciprocal Rank (MRR) to evaluate the ranking accuracy of the recommendations, which computes the average reciprocal rank of the first relevant item in the recommendation list. Binomial Diversity Framework (BinomDiv) [268] was used to evaluate the individual diversity, namely the degree of diversity in the recommendation lists based on item genres extracted from DBpedia.

6.4.3 Evaluated methods

We compared the performance of our proposed methods against the following baseline algorithms:

- **POP**. Non personalized baseline that always recommends the most popular items not yet liked by the user. Popularity is measured as the number of users in the dataset that liked the item.

- **UNN**. User-based nearest neighbors with Jaccard similarity. The size of the neighborhood is tuned for each dataset using a validation set.

- **INN**. Item-based nearest neighbors with Jaccard similarity and indefinite neighborhood size.

- **iMF**. Matrix factorization method for positive-only feedback [130] trained using the fast ALS technique by [213].

- **BPR**. Bayesian personalized ranking from implicit feedback [218]. We used for our experiments the implementation available in LibRec [113].

- **FISM**. Factored item similarity model by [138]. We used the implementation of the FISMauc variant optimized for the item ranking problem available in LibRec [113].

- **HeteRec**. Graph-based recommender system proposed in [288], based on a diffusion method of user preferences following different meta-paths.

- **SPRank**. Originally proposed in [194], it implements a hybrid approach to compute recommendations with LOD datasets. We used a publicly available implementation of SPRank[15].

With the exception of POP (which only uses target domain data) and SPRank, we considered the application of all the baselines to both single- and cross-domain scenarios. We were not able to compute meaningful results for SPRank by using DBpedia properties shown in Table 6.2 due to the structure of the connections between domains in the underlying knowledge graph. All the paths calculated by SPRank to link items in different domains resulted in being not very relevant thus bringing to shallow performances of the algorithm. Moreover, given the datasets adopted for the experimental evaluation, we were not able to generate all the meta-path needed to compute recommendations. We used machines with up to 3 TB of disk space but it was not sufficient.

Hereafter we use the prefix `CD-` to indicate that the algorithm is operating in cross-domain mode using the union of the rating matrices from the source and target domains. We did not consider for our evaluation the SemanticSVD++ method by [227], as it is designed for rating prediction rather than item ranking. Moreover, preliminary tests showed that its performance was much lower than the other methods, and that its training time was about one order of magnitude larger.

We evaluated the three methods presented in this study:

- **SimMF**. Our matrix factorization model regularized with similarity prediction described in 6.3.1.

[15] `https://github.com/sisinflab/lodreclib`

- **NeighborMF**. Our proposed matrix factorization model with neighborhood-based regularization from 6.3.2.

- **CentroidMF**. Our matrix factorization model from 6.3.3 that uses the neighbor's centroid to regularize the target domain item factors.

We tuned the hyperparameters of the considered recommendation models using a held-out validation set of likes, as we explain in the next section. For UNN, we only had to select the size of the user neighborhoods. For the matrix factorization models, in contrast, the number of hyperparameters is larger, namely, the dimensionality of the latent factor space k, the amount of regularization λ, and the confidence parameter for positive-only feedback α. Moreover, the models proposed also include the cross-domain regularization rate λ_C, which controls the contribution of the inter-domain item similarities. Finally, for NeighborMF and CentroidMF, we tuned the size n of the item neighborhoods $N(j)$, and the possibility to normalize the neighbors' similarities so that the sum to 1, as explained in 6.3.2.

The high number of parameters to tune rules out the possibility of performing a grid search for the best values. Hence, we used Bayesian Optimization techniques [252] that train Machine Learning models to predict candidate values that are likely to maximize a given function while simultaneously reducing the uncertainty of over unknown parameter values.

We tuned the parameters of the single-domain methods only on the target domain, and used the same values for their cross-domain variants. For UNN, the optimal number of neighbors was $n = 50$ for books, and $n = 100$ for movies and music. For iMF we obtained the optimal parameters $k = (10, 29, 21), \lambda = (10^{-5}, 0.823, 1)$, and $\alpha = (6, 7, 10)$ for books, movies, and music, respectively. For BPR we used $\lambda = 0.01$ for regularization and $\eta = 0.01$ as learning rate. In the case of FISM, we used $\lambda = 0.001$ and $\eta = 10^{-5}$. The optimal values for our proposed cross-domain models are reported in 6.5.

Table 6.5: Optimal hyperparameters for SimMF, NeighborMF, and CentroidMF. The last column indicates whether the similarities in the neighborhood are normalized or not.

	Source	**Method**	k	λ	α	λ_C	n	**Norm.**
Books	Movies	SimMF	112	0	1	10^{-8}		
		NeighborMF	134	1	1	9.125	49	✓
		CentroidMF	153	0.999	1	8.778	100	✓
	Music	SimMF	10	1	16	10^{-8}		
		NeighborMF	10	0	18	10	100	✓
		CentroidMF	10	0	14	0.109	100	
Movies	Books	SimMF	12	1	1	0.002		
		NeighborMF	12	1	1	10	81	✓
		CentroidMF	14	0.100	1	0.200	1	✓
	Music	SimMF	35	0	1	1.6×10^{-6}		
		NeighborMF	51	1	1	10	100	
		CentroidMF	29	1	1	9.494	99	✓
Music	Books	SimMF	10	1	1	0.039		
		NeighborMF	10	0.995	1	3.014	100	✓
		CentroidMF	10	0.724	1	1.673	14	
	Movies	SimMF	11	0.571	4	0.641		
		NeighborMF	10	0.978	2	0.699	46	
		CentroidMF	10	0.562	2	10	3	✓

6.4.4 Results

In this section we present the results of the conducted experiments to evaluate the proposed matrix factorization models. First, we analyze several semantic relatedness metrics to compute the inter-domain item similarities. Next, we report the ranking accuracy and diversity of the evaluated recommendation approaches, and study how the size and diversity of the source domain user profile impacts on the target recommendations.

6.4.5 Inter-domain item semantic similarity

The goal of our first experiment is to analyze the performance of several semantic relatedness metrics to compute the inter-domain similarities that we later exploit in our matrix factorization models. We considered the following strategies:

- TF-IDF. We use the semantically-enriched item profiles (see 6.4.1 to build TF-IDF vector profiles based on the metadata of each item. The similarity score between a source domain item and a target domain item is computed as the cosine of their corresponding TF-IDF vectors.

- ESA. The Explicit Semantic Analysis technique proposed by [103]. Instead of using the semantic metadata, we map each item to its corresponding Wikipedia article. Then, based on the text of the article, ESA extracts a set of other related Wikipedia articles, which represent semantic concepts, and builds a TF-IDF profile from the extracted concepts. Finally, the similarity score between two items is computed as the cosine of their corresponding concept-based vectors.

- M&W. The approach proposed by [175] computes the semantic relatedness between two items using the overlap of their sets of inlinks and outlinks in the Wikipedia hyperlink graph.

- Katz. Based on Katz's centrality measure, the relatedness between two items is computed as the accumulated probability of the top shortest paths between their corresponding entities in the semantic network [131].

Table 6.6: MRR of the evaluated semantic relatedness metrics.

Source	**Target**	**TF-IDF**	**ESA**	**M&W**	**Katz**
Books	Movies	0.058	0.030	**0.123**	0.092
	Music	0.028	0.015	**0.042**	0.022
Movies	Books	**0.054**	0.011	0.031	0.013
	Music	**0.030**	0.011	0.028	0.009
Music	Books	0.010	0.006	**0.052**	0.020
	Movies	0.013	0.018	**0.088**	0.006

We evaluated the previous semantic relatedness metrics indirectly by comparing their performance in the item recommendation task. For such purpose, we chose a content-based recommendation model with no parameters, so that we can fairly measure the effect of each similarity on the item ranking quality. According to this simple model, the relevance score of an item is computed as the accumulated similarity with the items in the user's profile:

$$s(u,i) = \sum_{j \in I(u)} s_{ij} \tag{6.18}$$

where s_{ij} is computed any of the methods described above.

The results of our experiment are shown in 6.6. For easier comparison according the methodology from 6.4.2, we averaged the MRR scores for all the cold start sizes in each source-target domain combination. We conclude from the table that M&W is the best performing metric, beating all the other approaches except when considering the movie domain as source, in which case it is still competitive. Hence, in the following experiments we evaluate our proposed matrix factorization models using M&W as the backing semantic similarity. Finally, we note that the low values for MRR are due to the simple recommendation algorithm chosen for this experiment.

6.4.6 Item ranking accuracy

In our second experiment we analyze the accuracy of the item rankings generated by the evaluated recommendation approaches. We aim to understand if cross-domain variants are in general more effective than single-domain ones, and whether the proposed matrix factorization models are able to outperform the other methods in cold start settings.

6.7 shows the ranking accuracy for book recommendations in terms of MRR. We report the average results for cold start user profiles from sizes 6–10, as we observed that in those cases the trends are stable and, in general, single-domain baselines start to be effective. We remark that, according the evaluation methodology described in 6.4.2, the number of test users remains constant regardless of the profile size, which we control by iteratively downsampling the training portion of their profile (see 6.1).

We notice from the table that, in general, approaches exploiting cross-domain movies or music preferences provide better recommendations than their single-domain counterparts. In case auxiliary movie preferences are available, we observe that the proposed NeighborMF and CentroidMF models achieve the best performance when only 1–3 book likes are observed. Moreover, in that case, our cross-domain matrix factorization models perform much better than the single-domain baselines. However, once 4 likes are available, CD-INN and single-domain HeteRec are more effective approaches. When the auxiliary preferences consist of music likes, we see that CD-INN is the overall best method, although it is only useful for profiles of size 1. For larger profiles, it is better to use single-domain baselines than any cross-domain method that uses music preferences. In summary, we conclude that music preferences are not useful for book recommendations, whereas movie likes could be used to improve the performance, specially with NeighborMF and CentroidMF for 1–3 book likes. We observe the bad performance of SPRank in cold start situations compared to the other baselines.

In 6.8 we show the results for movie recommendations. We observe that most of the cross-domain approaches are able to provide recommendations better than the most popular items for completely new movie users, and that CD-HeteRec is clearly the best performing approach. If the auxiliary cross-domain data consists of book preferences, we notice that the proposed matrix factorization models outperform the best single-domain baselines. However, in this situation CD-INN is even a better method, clearly providing more accurate recommendations than any other approach

Table 6.7: Accuracy (MRR) for cold start users in the target books domain. The three groups of rows correspond to single-domain, cross-domain with movies as source, and cross-domain with music as source, respectively. Best values for each single- and cross-domain configuration are shown in bold.

		Number of book *likes*						
	Method	0	1	2	3	4	5	6–10
	POP	**0.242**	**0.244**	0.246	0.248	0.251	0.252	0.260
	UNN		0.222	**0.265**	**0.286**	0.289	0.290	0.322
	INN		0.145	0.177	0.216	0.241	0.262	0.316
	iMF		0.171	0.194	0.235	0.255	0.271	0.301
	BPR		0.110	0.116	0.136	0.154	0.157	0.193
	FISM		0.228	0.230	0.234	0.234	0.238	0.245
	HeteRec		0.218	0.244	0.279	**0.297**	**0.316**	**0.351**
	SPRank		0.048	0.055	0.070	0.065	0.062	0.059
Movies	CD-UNN	0.186	0.148	0.170	0.175	0.189	0.190	0.212
	CD-INN	0.262	0.265	0.275	0.291	**0.301**	**0.307**	**0.339**
	CD-iMF	0.261	0.262	0.268	0.272	0.275	0.274	0.287
	CD-BPR	0.217	0.200	0.218	0.237	0.235	0.238	0.251
	CD-FISM	0.235	0.228	0.225	0.231	0.236	0.235	0.245
	CD-HeteRec	**0.264**	0.248	0.261	0.268	0.278	0.277	0.298
	SimMF	0.253	0.268	0.274	0.284	0.289	0.290	0.296
	NeighborMF	0.253	**0.272**	0.282	**0.294**	0.293	0.293	0.301
	CentroidMF	0.252	0.271	**0.283**	0.289	0.293	0.295	0.301
Music	CD-UNN	0.136	0.103	0.115	0.120	0.138	0.140	0.157
	CD-INN	0.259	0.260	**0.266**	**0.278**	**0.296**	**0.302**	**0.329**
	CD-iMF	0.259	**0.261**	0.262	0.264	0.266	0.270	0.282
	CD-BPR	0.218	0.199	0.199	0.216	0.228	0.228	0.250
	CD-FISM	0.230	0.228	0.227	0.229	0.236	0.233	0.245
	CD-HeteRec	**0.266**	0.249	0.251	0.259	0.270	0.267	0.281
	SimMF	0.255	0.259	0.258	0.264	0.268	0.273	0.281
	NeighborMF	0.253	0.258	0.258	0.263	0.267	0.273	0.280
	CentroidMF	0.255	0.259	0.260	0.264	0.267	0.273	0.281

Table 6.8: Accuracy (MRR) for cold start users in the target movies domain. The three groups of rows correspond to single-domain, cross-domain with books as source, and cross-domain with music as source, respectively. Best values for each single- and cross-domain configuration are shown in bold.

		Number of movie *likes*						
	Method	0	1	2	3	4	5	6–10
	POP	**0.285**	0.287	0.289	0.292	0.294	0.297	0.305
	UNN		**0.332**	0.320	0.318	0.330	0.348	0.405
	INN		0.233	0.300	0.336	0.359	**0.377**	**0.413**
	iMF		0.256	0.291	0.314	0.334	0.348	0.388
	BPR		0.225	0.256	0.276	0.299	0.315	0.350
	FISM		0.257	0.265	0.263	0.266	0.267	0.270
	HeteRec		0.315	**0.346**	**0.357**	**0.366**	0.374	0.395
	SPRank		0.107	0.131	0.139	0.142	0.140	0.150
Books	CD-UNN	0.219	0.169	0.185	0.219	0.256	0.292	0.385
	CD-INN	0.344	**0.347**	**0.371**	**0.386**	**0.398**	**0.410**	**0.435**
	CD-iMF	0.267	0.298	0.325	0.347	0.365	0.377	0.413
	CD-BPR	0.018	0.189	0.237	0.254	0.278	0.298	0.326
	CD-FISM	0.338	0.267	0.263	0.283	0.273	0.287	0.282
	CD-HeteRec	**0.479**	0.320	0.349	0.359	0.367	0.375	0.396
	SimMF	0.328	0.334	0.348	0.361	0.371	0.382	0.409
	NeighborMF	0.330	0.335	0.348	0.361	0.371	0.383	0.409
	CentroidMF	0.329	0.332	0.346	0.359	0.371	0.378	0.408
Music	CD-UNN	**0.387**	0.282	0.305	0.320	0.334	0.348	0.383
	CD-INN	0.342	0.347	0.353	0.359	0.365	0.371	0.390
	CD-iMF	0.301	0.326	0.344	0.362	0.374	0.385	0.418
	CD-BPR	0.352	0.305	0.316	0.332	0.332	0.343	0.361
	CD-FISM	0.105	0.089	0.093	0.089	0.091	0.091	0.093
	CD-HeteRec	0.367	0.336	0.344	0.350	0.355	0.360	0.374
	SimMF	0.339	0.351	0.361	0.374	0.384	0.396	0.419
	NeighborMF	0.353	**0.364**	**0.374**	**0.385**	**0.394**	**0.404**	**0.427**
	CentroidMF	0.345	0.355	0.367	0.377	0.385	0.395	0.418

from profile sizes 1–10. This is due to the high degree of overlap between the users of books and movies domains (79.7%, see 6.4), which allows CD-INN to compute very accurate item similarities based on the patterns of likes. Instead, when the source domain contains music preferences, we see that NeighborMF, CentroidMF, and SimMF, in that order, are consistently the best performing approaches for sizes 1–10. By regularizing item factors independently, NeighborMF is able to transfer source domain knowledge more effectively, which we also note is due to the greater contribution of cross-domain information (larger values of λ_C in 6.5). In summary, both book and music preferences are helpful for cold start movie recommendations, while our models are more effective when exploiting auxiliary music likes. On a side note, we observe the better performance of UNN over POP on the single domain setting when only 1 like is available. Looking at the results, we found that this is caused by the Jaccard-based similarity, which favors neighbors with small profiles that have rated similar items with high probability. A discussion of this phenomenon is outside of the line of research, and we refer the reader to [33] for a detailed explanation. HeteRec, on the other hand, exploits additional information from item metadata to compute more accurate recommendations than POP while SPRank confirms its bad behavior in cold start scenarios.

Finally, the results for music recommendations are shown in 6.9. As previously, CD-HeteRec is a very good performing approach to provide recommendations for completely new users, in both cross-domain configurations. Once 2 music likes are available, CD-INN is clearly the most competitive approach, independently of the used source domain. Again, we argue that this is due to the high number of music users who also have book and movie preferences, which allows CD-INN to compute very accurate rating-based similarities for items (see last column of 6.4). However, when the source domain consists of book preferences, we see that the proposed NeighborMF and CentroidMF models are slightly better than other cross-domain approaches if only 1 music like is provided. Anyway, even better performance can be achieved in this case simply using the single-domain UNN baseline, which does not need any extra information. Hence, single-domain baselines are compelling approaches for cold start music recommendations, and even though the proposed

Table 6.9: Accuracy (MRR) for cold start users in the target music domain. The three groups of rows correspond to single-domain, cross-domain with books as source, and cross-domain with movies as source, respectively. Best values for each single- and cross-domain configuration are shown in bold.

		Number of music *likes*						
	Method	0	1	2	3	4	5	6–10
	POP	**0.335**	0.337	0.340	0.342	0.345	0.347	0.354
	UNN		**0.422**	0.389	0.389	0.419	0.448	**0.517**
	INN		0.320	0.391	0.426	**0.455**	**0.474**	**0.517**
	iMF		0.347	**0.396**	**0.427**	0.451	0.471	**0.517**
	BPR		0.330	0.377	0.409	0.432	0.450	0.488
	FISM		0.096	0.100	0.100	0.100	0.101	0.100
	HeteRec		0.358	0.395	0.421	0.442	0.463	0.510
	SPRank		N/A	N/A	N/A	N/A	N/A	N/A
Books	CD-UNN	0.290	0.244	0.266	0.300	0.344	0.387	0.487
	CD-INN	0.310	0.368	**0.416**	**0.442**	**0.465**	**0.482**	**0.522**
	CD-iMF	0.200	0.330	0.391	0.423	0.451	0.471	0.518
	CD-BPR	0.004	0.267	0.323	0.362	0.380	0.404	0.433
	CD-FISM	0.153	0.124	0.105	0.126	0.116	0.118	0.113
	CD-HeteRec	**0.514**	0.367	0.407	0.432	0.453	0.474	0.516
	SimMF	0.310	0.368	0.401	0.424	0.446	0.461	0.496
	NeighborMF	0.328	**0.372**	0.402	0.425	0.445	0.461	0.496
	CentroidMF	0.325	0.370	0.402	0.425	0.444	0.461	0.496
Movies	CD-UNN	0.435	0.274	0.306	0.336	0.369	0.400	0.484
	CD-INN	0.412	**0.431**	**0.451**	**0.467**	**0.478**	**0.490**	**0.522**
	CD-iMF	0.293	0.356	0.398	0.428	0.454	0.474	0.516
	CD-BPR	0.431	0.313	0.351	0.391	0.402	0.413	0.448
	CD-FISM	0.093	0.061	0.069	0.067	0.070	0.071	0.064
	CD-HeteRec	**0.515**	0.406	0.426	0.442	0.451	0.464	0.495
	SimMF	0.361	0.393	0.420	0.438	0.455	0.467	0.500
	NeighborMF	0.353	0.385	0.409	0.429	0.445	0.458	0.494
	CentroidMF	0.354	0.386	0.413	0.431	0.447	0.460	0.495

models are able to improve the quality of the item rankings by exploiting cross-domain item metadata, CD-INN, which is purely based on patterns of likes, is the best performing approach.

6.4.7 Recommendation diversity

In this subsection we analyze the diversity of the recommendation lists generated by the methods, as an alternative dimension of ranking quality.

6.10 shows the diversity of book recommendations in terms of the Binomial Diversity metric at cutoff 10 (BinomDiv@10). We observe that, in general, cross-domain approaches provide more diverse recommendations than their single-domain counterparts. However, we note several differences with respect to the accuracy results reported in 6.7. First, CD-UNN is consistently the superior algorithm in terms of diversity, whereas its accuracy results were the poorest among single- and cross-domain approaches. Second, when the source domain consists of movie likes, our proposed models achieve slightly worse diversity than other cross-domain approaches, specially for book profile sizes between 1–3 likes. This is in contrast with the results obtained in 6.7, where our methods performed best precisely in that range. We conclude that there is a clear trade-off between recommendation accuracy and diversity, and that the metric of interest depends on the particular application domain. We argue, however, that in cold start situations providing relevant suggestions may be more useful than recommending diverse, but not relevant items, if the ultimate goal of a system is to keep new users engaged.

The diversity results for movie recommendations are summarized in 6.11. We see that CD-FISM, CD-BPR, and CD-UNN provide the most diverse yet not relevant recommendations. Comparing the sources of auxiliary user preferences, we note that the diversity of the cross-domain baselines is roughly the same as their single-domain versions (comparing, e.g., HeteRec and CD-HeteRec) when considering book likes. In contrast, if the source domain contains music likes their diversity is significantly hurt. By comparing these results with 6.8 we observe once again the accuracy-diversity trade-off. Most methods' MRR greatly benefits from additional music likes at the expense of worse diversity. The exception is CD-FISM,

Table 6.10: Diversity (BinomDiv@10) for cold start users in the books domain. The three groups of rows correspond to single-domain, cross-domain with movies as source, and cross-domain with music as source, respectively. Best values for each single- and cross-domain configuration are shown in bold.

		Number of book *likes*						
	Method	0	1	2	3	4	5	6–10
	POP	**0.739**	0.674	0.690	0.702	0.703	0.710	**0.736**
	UNN		**0.733**	**0.706**	**0.716**	**0.709**	**0.729**	0.715
	INN		0.655	0.674	0.654	0.665	0.672	0.669
	iMF		0.583	0.606	0.630	0.645	0.657	0.664
	BPR		0.696	0.700	0.715	0.690	0.698	0.696
	FISM		0.513	0.692	0.708	0.686	0.706	0.719
	HeteRec		0.609	0.623	0.653	0.672	0.680	0.693
	SPRank		0.121	0.129	0.145	0.157	0.157	0.150
Movies	CD-UNN	**0.792**	**0.833**	**0.816**	**0.791**	**0.778**	**0.784**	**0.746**
	CD-INN	0.740	0.676	0.683	0.684	0.680	0.692	0.695
	CD-iMF	0.724	0.660	0.674	0.689	0.686	0.686	0.702
	CD-BPR	0.514	0.458	0.484	0.425	0.454	0.465	0.471
	CD-FISM	0.453	0.464	0.479	0.488	0.495	0.499	0.533
	CD-HeteRec	0.747	0.673	0.672	0.680	0.690	0.704	0.709
	SimMF	0.702	0.649	0.671	0.676	0.682	0.690	0.706
	NeighborMF	0.690	0.652	0.660	0.671	0.680	0.682	0.702
	CentroidMF	0.699	0.647	0.659	0.668	0.684	0.686	0.702
Music	CD-UNN	0.744	**0.811**	**0.797**	**0.771**	**0.746**	**0.734**	**0.731**
	CD-INN	**0.746**	0.676	0.683	0.684	0.674	0.689	0.691
	CD-iMF	0.720	0.657	0.664	0.674	0.690	0.692	0.696
	CD-BPR	0.303	0.333	0.374	0.384	0.358	0.374	0.380
	CD-FISM	0.345	0.362	0.382	0.405	0.406	0.424	0.466
	CD-HeteRec	0.744	0.668	0.655	0.665	0.676	0.687	0.693
	SimMF	0.724	0.656	0.675	0.684	0.692	0.692	0.708
	NeighborMF	0.721	0.657	0.674	0.684	0.690	0.693	0.709
	CentroidMF	0.721	0.655	0.673	0.681	0.692	0.690	0.705

Table 6.11: Diversity (BinomDiv@10) for cold start users in the movies domain. The three groups of rows correspond to single-domain, cross-domain with books as source, and cross-domain with music as source, respectively. Best values for each single- and cross-domain configuration are shown in bold.

		Number of movie *likes*						
	Method	0	1	2	3	4	5	6–10
	POP	**0.401**	0.304	0.336	0.354	0.368	0.378	0.399
	UNN		0.360	0.385	0.404	0.392	0.396	0.394
	INN		0.289	0.308	0.315	0.321	0.323	0.332
	iMF		0.299	0.320	0.335	0.344	0.347	0.362
	BPR		**0.590**	0.608	**0.628**	0.644	**0.650**	**0.653**
	FISM		0.561	**0.614**	0.594	**0.689**	0.636	0.644
	HeteRec		0.311	0.328	0.334	0.337	0.341	0.348
	SPRank		0.218	0.242	0.260	0.262	0.254	0.269
Books	CD-UNN	0.467	0.509	0.479	0.446	0.425	0.414	0.397
Books	CD-INN	0.327	0.291	0.314	0.323	0.329	0.331	0.339
Books	CD-iMF	0.341	0.294	0.317	0.327	0.333	0.338	0.350
Books	CD-BPR	**0.646**	**0.677**	0.645	**0.668**	**0.624**	**0.643**	**0.666**
Books	CD-FISM	0.548	0.549	**0.671**	0.574	0.609	0.625	0.664
Books	CD-HeteRec	0.316	0.310	0.328	0.335	0.337	0.341	0.348
Books	SimMF	0.308	0.265	0.297	0.307	0.320	0.325	0.339
Books	NeighborMF	0.315	0.266	0.298	0.306	0.321	0.325	0.338
Books	CentroidMF	0.313	0.273	0.302	0.315	0.326	0.334	0.348
Music	CD-UNN	0.368	0.404	0.386	0.376	0.373	0.372	0.376
Music	CD-INN	0.309	0.240	0.268	0.283	0.297	0.304	0.321
Music	CD-iMF	0.270	0.231	0.270	0.289	0.302	0.315	0.332
Music	CD-BPR	0.372	0.439	0.411	0.438	0.446	0.445	0.476
Music	CD-FISM	**0.653**	**0.720**	**0.705**	**0.499**	**0.645**	**0.732**	**0.688**
Music	CD-HeteRec	0.333	0.271	0.298	0.314	0.324	0.333	0.349
Music	SimMF	0.311	0.254	0.288	0.303	0.317	0.324	0.340
Music	NeighborMF	0.311	0.259	0.290	0.308	0.320	0.329	0.344
Music	CentroidMF	0.302	0.246	0.279	0.297	0.310	0.319	0.338

which follows the opposite trend: source music likes lead to significantly worse accuracy but improved diversity. We leave for future work an analysis of CD-FISM to understand which of its characteristics causes this behavior. Finally, we remark the good performance of the NeighborMF method when source music likes are exploited, as it is able to provide a good trade-off of decent diversity and the most accurate recommendations (see 6.8).

Last, we report the diversity results for music recommendations in 6.12. Once again, CD-FISM, which achieved the poorest accuracy in 6.9, provides the most diverse recommendations for all music profile sizes in the 1–10 range. However, for completely new users, we highlight the very good performance of CD-HeteRec, which not only is able to generate diverse recommendations, but also achieved the best accuracy results in terms of MRR. The remaining cross-domain approaches are in general worse than single-domain UNN, independently of the exploited source domain. It is also worth noting the contrasting results for CD-INN. While it provides the best performance in terms of accuracy (see 6.9), its diversity is the worst for books and only average for movies.

In summary, we observe a clear trade-off between accurate and diverse recommendations. In general, when approaches perform well in terms of MRR they tend to suffer in terms of diversity, and vice versa.

6.5 Conclusions and future work

Collaborative filtering approaches have become the most investigated and popular solutions to the cross-domain recommendation problem, as they only mine patterns of user-item preferences (i.e., ratings), and do not require any information about the content of the items to bridge the domains of interest. Some other approaches, however, have shown that content-based relations (e.g., based on social tags) can be exploited to bridge the domains more effectively. In this context, recent initiatives such as the Linked Open Data project provide large interconnected repositories of structured knowledge than can be exploited to relate multiple types of data. Such heterogeneous networks allow establishing content-based links between different

Table 6.12: Diversity (BinomDiv@10) for cold start users in the music domain. The three groups of rows correspond to single-domain, cross-domain with books as source, and cross-domain with movies as source, respectively. Best values for each single- and cross-domain configuration are shown in bold.

		Number of music *likes*						
	Method	0	1	2	3	4	5	6–10
	POP	**0.324**	0.228	0.262	0.282	0.295	0.305	0.326
	UNN		0.296	0.332	0.348	0.347	0.330	0.306
	INN		0.200	0.213	0.219	0.223	0.229	0.236
	iMF		0.196	0.217	0.232	0.241	0.249	0.259
	BPR		0.539	0.577	0.589	0.590	0.594	0.619
	FISM		**0.683**	**0.766**	**0.709**	**0.731**	**0.737**	**0.676**
	HeteRec		0.227	0.264	0.280	0.288	0.296	0.304
	SPRank		N/A	N/A	N/A	N/A	N/A	N/A
Books	CD-UNN	0.325	0.429	0.414	0.393	0.366	0.346	0.314
	CD-INN	0.269	0.215	0.227	0.232	0.235	0.240	0.244
	CD-iMF	0.270	0.214	0.233	0.240	0.249	0.252	0.258
	CD-BPR	0.570	0.585	0.578	0.602	0.592	0.603	0.609
	CD-FISM	**0.607**	**0.597**	**0.677**	**0.648**	**0.774**	**0.726**	**0.674**
	CD-HeteRec	0.295	0.233	0.271	0.286	0.294	0.302	0.309
	SimMF	0.274	0.220	0.240	0.249	0.257	0.264	0.275
	NeighborMF	0.254	0.220	0.241	0.251	0.259	0.265	0.275
	CentroidMF	0.253	0.218	0.238	0.249	0.257	0.263	0.273
Movies	CD-UNN	0.296	0.411	0.380	0.358	0.347	0.329	0.312
	CD-INN	0.277	0.231	0.255	0.264	0.270	0.272	0.275
	CD-iMF	0.248	0.229	0.254	0.264	0.271	0.272	0.277
	CD-BPR	0.476	0.515	0.526	0.504	0.526	0.529	0.545
	CD-FISM	**0.601**	**0.664**	**0.541**	**0.757**	**0.578**	**0.771**	**0.669**
	CD-HeteRec	0.372	0.271	0.314	0.331	0.342	0.349	0.360
	SimMF	0.225	0.207	0.239	0.250	0.259	0.264	0.278
	NeighborMF	0.252	0.226	0.251	0.265	0.269	0.274	0.283
	CentroidMF	0.264	0.233	0.257	0.270	0.274	0.279	0.286

types of items, and thus providing a new mechanism to bridge domains for cross-domain recommendation.

In this study, we have exploited Linked Open Data to extract metadata about items in three recommendation domains. Using this additional information, we were able to find relations between items in different domains, and ultimately compute inter-domain item similarities. This could be a limit of the presented approaches whenever the underlying LOD knowledge graph does not expose semantic links between items in different domains, e.g., when the source and target domains do not share information, i.e., there is no direct or indirect link between items in different domains or it is not possible to link an item in the catalog to the corresponding entity in the knowledge graph. In fact, in these cases, it is not possible to compute pairwise semantic similarity values between items belonging to different domains.

We then proposed three novel matrix factorization models for cross-domain recommendation that exploit the computed similarities to link knowledge across domains. Experiments in cold start scenarios showed that depending on the involved source and target domains, cross-domain recommendations exploiting item metadata can be more accurate for users with few preferences in the target domain. However, the improved accuracy comes at the cost of less diversity among the recommendations, and approaches thriving in diversity tend to be less accurate. We argue, nonetheless, that in cold start the priority of a system may be keeping the user engaged by delivering relevant recommendations rather than diverse, non relevant ones.

Regarding the categorization presented in 6.2.1, the models proposed in this study belong to the category of **knowledge linkage** cross-domain recommendation approaches. We applied our approaches to the **linked-domain exploitation** task with the goal of **addressing the user cold start** problem. In addition to the results reported here, we conjecture that item metadata may be prove more useful in cross-domain scenarios with low user overlap. In these cases, approaches purely based on collaborative filtering are likely to struggle to compute accurate item-item similarities. Moreover, in our work we relied on advanced Bayesian Optimization techniques to find the optimal hyperparameters of the models, and in particular the values of the cross-domain regularization λ_C and the item neighborhood size n parameters. It would be interesting, however, to analyze the performance of the models in terms of these parameters to better understand the importance of auxiliary information. We did not report these results due to the high number of possible combinations of different parameter values, source-target domain configurations, cold start profile sizes, and cross-validation folds, which may make it very difficult to extract conclusions that consistently hold trough all the possible scenarios.

Chapter 7

Interpretability of Factorization Machines

7.1 Introduction

Research on transparency, and interpretability of predictive models is gaining momentum since it has been recognized as a key element in the next generation of recommendation algorithms. Providing explanations may increase the user awareness in the decision-making process leading to fast (efficiency), conscious and right (effectiveness) decisions. When equipped with interpretablity of recommendation results, a system ceases to be just a black-box (transparency) [249, 262, 290] and users are more willing to extensively exploit the predictions [261, 122]. Transparency increases their trust [91] (also exploiting specific semantic structures [86]), and satisfaction in using the system. For a recommender system, the user's trust is becoming more and more important since it leads to better performance [159]. Most of the proposed approaches to interpret recommendation results seek to show how they are related to users' preferences. In a nutshell, we may say that interpretations for recommendation results can be item-based, user-based or feature-based. Item-based interpretations make use of the shared set of items among users [2]; User-based explanations rely on sets of most similar users, like in [122]; Feature-based explanations exploit features of recommended items as director, genre, and cast [261]. Among interpretable models, we may distinguish between those based on Content-based (CB) approaches and those based on Collaborative filtering (CF) ones. CB algorithms provide recommendations by exploiting the available content and matching it with a user profile [209, 68]. The use of content features makes the model interpretable even though attention has to be paid since a CB approach "*lacks serendipity and requires extensive manual efforts to match the user interests to content profiles*" [296]. It is worth noticing that these features can be dramatically different depending on the considered scenario: a movie recommendation could be based on the director, actors, the producer, the genre whereas a book recommendation may be explained by the author, the book formats or the saga. Sometimes, this prevents the straight adoption of a model independently of the addressed knowledge domain.

When content is missing, the recommender may rely only on the relationships between users and the rates they provide to items in a collaborative fashion. Based on the different collaborative approaches, these relationships may focus on items or users. Interpretation of CF results will then inevitably reflect the approach adopted by the algorithm. For instance, an item-based and a user-based recommendation could be explained, respectively, as "other users who have experienced A have experienced B" or "similar users have experienced B". Concerning collaborative approaches to recommendation, when we use a kNN approach, either item-based or user-based, we explicitly refer to users and items in the system. Unfortunately, things change when we adopt more powerful and accurate Deep Learning [60], or model-based algorithms and techniques for the computation of a recommendation list. Such approaches project items and users in a new vector space of latent features [150] thus making the final result not directly interpretable. Indeed, it is possible to compute items and users similarities via latent factor exploitation, but we lose entirely any reference to the original user-item interaction and then to an explicit justification for the computed recommendations.

In the last years, many approaches have been proposed that take advantage of side information to enhance the performance of latent factor models. Side information can refer to items as well as users [276] and can be either structured [255] or semi-structured [297, 25, 62]. Interestingly, in [296] the authors argue about a new generation of knowledge-aware recommendation engines able to exploit information encoded in knowledge graph (KG) to produce meaningful recommendations: "*For example, with knowledge graph about movies, actors, and directors, the system can explain to the user a movie is recommended because he has watched many movies starred by an actor*".

In this work, we propose a `knwowledge-aware Hybrid Factorization Machine` (`kaHFM`) to train interpretable models in recommendation scenarios. `kaHFM` relies on Factorization Machines [216] and it extends them in different key aspects making use of the semantic information encoded in a knowledge graph.

Interpretability without accuracy of results is not sufficient when designing a recommendation engine since it is unlikely that an inaccurate model will be adopted. It would be highly beneficial for users to develop recommender systems that are accurate, and, at the same time, that are built upon an interpretable technique, so that the learned model can be, in case, exploited to generate explanations. In this direction, we show how `kaHFM` may exploit data coming from knowledge graphs as side information to build a recommender system whose final results are accurate and, at the same time, semantically interpretable. With `kaHFM` we build a model in which the meaning of each latent factor is bound to an explicit content-based feature extracted from a knowledge graph. Doing this, after the model has been trained, we still have an explicit reference to the original semantics of the features describing the items, thus making possible the interpretation of the final results. Interestingly, we will see that the explicit mapping of latent features to content-based ones makes possible to exploit characteristics of these latter to implement a more effective initialization technique.

We evaluated `kaHFM` on six different publicly available datasets by getting content-based explicit features from data encoded in the DBpedia[1] knowledge graph. We analyzed the performance of the approach in terms of accuracy, diversity, and novelty of results by exploiting categorical, ontological and factual features (see Section 7.3.1). For each of them, we used public mappings to DBpedia. Finally, we tested the robustness of kaHFM with respect to its interpretability showing that it ranks meaningful features higher and is able to regenerate them in case they are removed from the original dataset.

With kaHFM we address the following research questions:

RQ1 Can we develop a model-based recommendation engine whose results are very accurate and, at the same time, can be interpreted with respect to an explicitly stated semantics coming from a knowledge graph?

RQ2 Can we evaluate that the original semantics of items features is preserved after the model has been trained?

RQ3 How to measure with an offline evaluation that the proposed model is really able to identify meaningful features by exploiting their explicit semantics?

This investigation can be then summarized as:

- presentation of kaHFM: a framework that exploits data available in knowledge graphs to build semantically interpretable models for recommendation tasks;
- an experimental evaluation based on six different datasets to assess the performance of kaHFM in terms of accuracy, diversity, and novelty of computed results;
- introduction of two metrics, Semantic Accuracy (SA@K) and Robustness (n-Rob@K), to measure the interpretability of a knowledge-aware recommendation engine.

The remainder of the chapter is structured as follows: in the next section, we introduce the background technologies behind kaHFM and then we detail the overall approach in Section 7.3. The following section is devoted to the introduction of

[1] http://dbpedia.org

the two metrics we propose to assess the quality of `kaHFM` results in terms of interpretability. In Section 7.4 we describe the experimental setting while in Section 7.2 we report on related work. Conclusions and future lines of research close the chapter.

7.2 Related Work

In recent years, several explainable recommendation models that exploit matrix factorization have been proposed. It is well-known that one of the main issues of matrix factorization methods is that they are not easily explainable (since latent factors meaning is basically unknown). One of the first attempts to overcome this problem was proposed in [294, 297]. In this work, the authors propose Explicit Factor Model (EFM). Products' features and users' opinions are extracted with phrase-level sentiment analysis from users' reviews to feed a matrix factorization framework. Each latent factor is thus mapped with a particular explicit feature. After that, a few improvements to EFM have been proposed to deal with temporal dynamics [298] and to use tensor factorization [62]. In particular, in the latter the aim is to predict both user preferences on features (extracted from textual reviews) and items. This is achieved by exploiting the Bayesian Personalized Ranking (BPR) criterion [218]. Eventually, these preferences are combined to produce recommendation lists. We considered this work interesting because they also adopt a pair-wise learning to rank algorithm, but this is really different from ours since we exploited BPR to explicitly train the feature vectors to rank items. Further advances in MF-based explainable recommendation models have been proposed with Explainable Matrix Factorization (EMF) [1] in which the generated explanations are based on a neighborhood model. We differ from EMF as they do not take advantage of any external data source, other than they use a completely different model. Moreover, they introduced an additional regularizer into the factorization model to constrain users' vectors training. Similarly, in [2] an explainable Restricted Boltzmann Machine model has been proposed. It learns a network model (with an additional visible layer) that takes into account a degree of explainability. To measure it, they defined an ad-hoc *Explain-*

ability Score. Finally, an interesting work took advantage of sentiment analysis of users' reviews. Authors incorporate the sentiments and ratings into a matrix factorization model. The overall approach, named Sentiment Utility Logistic Model (SULM) [25], generates a novel kind of explanations composed by both items and features. In [215] recommendations are computed by generating and ranking personalized explanations in the form of explanation chains. OCuLaR [271] provides interpretable recommendations from positive examples based on the detection of co-clusters between users (clients) and items (products). The recommendation comes with the corresponding user-item co-clusters, which provide much more detailed information than usual collaborative-based explanations. In [129] authors propose a Multi Level Attraction Model (MLAM) in which they build two attraction models, for cast and story. Moreover, the story model is built upon two attraction models: for story level and for sentence level. The interpretability of the model is then provided in terms of attractiveness of Sentence level, Word level, and Cast member. In [210] the authors train a matrix factorisation model to complete the $U \times I$ matrix. They then use the complete (approximated) ratings matrix to compute a set of association rules that explain the obtained recommendations. In [76] the authors prove that, given the conversion probabilities for all actions of customer features, it is possible to transform the original historical data to a new space in order to compute a set of interpretable recommendation rules.

The design of a new explainable model is useless if it does not match with any existing explanations generation technique. For this reason we deepened the different ways of generating feature-based explanations in order to provide a flexible but accurate model. Probably the most used style of explanations of this kind are the content-based ones [256]. In their most simple form, authors consider similarities between items by taking into account both item properties and user ratings. Among the works that exploit content information to produce explainable recommendations, Tagsplanations [270] is worth to mention. It is fed by community tags and it exploits a relevance measure to weight tags w.r.t. items and user preferences. Furthermore, also demographic-based recommendations explanations have been inspected [300], in order to recommend items for particular types (age, location, gender) of users. The core of our model is a general Factorization Machines (FM) model [216]. Nowadays FMs are the most widely used factorization models because they offer a number of advantages w.r.t. other latent factors models such as SVD++ [148], PITF [221], FPMC [219]. First of all, FMs are designed for a generic prediction task while the others can be exploited only for specific tasks. Moreover it is a linear model and parameters can be estimated accurately even in high data sparsity scenarios. Nevertheless, several improvements have been proposed for FMs. For instance Neural Factorization Machines [120] have been developed to fix the inability of classical FMs to capture non linear structure of real-world data. This goal is achieved by exploiting the non linearity of neural networks. Furthermore, Attentional Factorization Machines [285] have been proposed that use an attention network to learn the importance of feature interactions. Finally, FMs have been specialized to better work as Context-Aware recommender systems [220]. Usually only top recommended items are provided to the user as suggestions as it is not feasible that a user analyze hundreds of recommended items. For this reason, ranking has become a much more important task than rating prediction [172]. This came with the development of a new class of learning algorithms (in which sorting correctly the recommendation list becomes the key task). These Learning to Rank [57] algorithms can be further categorized in Point-wise [151], Pair-wise [218, 169] and List-wise [245, 244]. In particular, Pair-wise approaches are usually considered as a good trade-off between ordering performances and computational complexity. Among this class of algorithms, Bayesian Personalized Ranking (BPR) [218] is one of the most widely adopted. It is based on a simple stochastic gradient descent algorithm to learn the relative order between positive items (items that a user has experienced in his past history) and negative items (items never rated by the user). BPR can be easily applied to Matrix Factorization and Factorization Machines (as in our work and in [26]).

7.3 Approach

For the sake of completeness, in this section we briefly recap the main technologies we adopted to develop `kaHFM`. We introduce Vector Space Models for recommender systems and then we give a quick overview on knowledge graphs and their Linked (Open) Data implementation.

Content-based recommender systems rely on the assumption that it is possible to predict the future behavior of users based on their personalized profile. User profiles can be built by exploiting the characteristics of the items they liked in the past or some other available side information. Several approaches have been proposed, that take advantage of side information in different ways: some of them consider tags [270], demographic data [300] or they extract information from collective knowledge bases [190]. Many of the most popular and adopted CB approaches make use of a Vector Space Model (VSM). In VSM users and items are represented by means of Boolean or weighted vectors in the same space. Their respective positions and the distance, or better the proximity, between them, provides a measure of how these two entities are related or similar. The space containing users and items vectors is composed by dimensions that may refer to characteristics of the users (e.g., the propensity to watch movies in the morning) or of the items as well as to their combination.

The choice of item features may substantially differ depending on their availability and application scenario: crowd-sourced tags, categorical, ontological, or textual knowledge are just some of the most exploited ones.

To sum up, in a CB approach we need (i) to get reliable items descriptions, (ii) a way to measure the strength of each feature for each item, (iii) to represent users and finally (iv) to measure similarities. Regarding the first point, nowadays we can easily get descriptions related to an item from the Web. In particular, thanks to the Linked Open Data initiative a lot of semantically structured knowledge is publicly available in the form of Linked Data datasets.

7.3.1 Knowledge Graphs and Linked Data

In 2012, Google announced its Knowledge Graph[2] (KG) as a new tool to improve the identification and retrieval of entities in return to a search query. Most of the knowledge encoded in Google Knowledge Graph actually came from Freebase which was a crowdsourced effort to create a base of facts in all possible knowledge domains. Alongside with the development of the above mentioned initiatives, following the original idea of a Semantic Web [38], new technologies have been developed and released with the aim of embedding structured knowledge with unambiguous semantics into Web pages in order to allow software agents to consume and elaborate information in an automated way. The original idea has been modified over the years thus making possible the creation of a full stack of semantic technologies and, more remarkably, gave birth to the Linking Open Data initiative[3] where a community of researchers and practitioners devoted an enormous effort to build publicly available knowledge bases of machine-understandable data. A knowledge base exploiting Semantic Web technologies is then represented through a graph (knowledge graph) in which entities are linked to each other by binary relations. In order to manage this knowledge model, several technologies and languages have been developed and among them the basic one is `Resource Description Framework` (RDF). It provides a simple graph-based data model to encode knowledge in a structured way by means of a triple $\langle \sigma, \rho, \omega \rangle$. In a knowledge graph, each triple represents the connection $\sigma \xrightarrow{\rho} \omega$ between two nodes, named *subject* (σ) and *object* (ω), through the *relation* (*predicate*) ρ. In an RDF knowledge graph, we usually find different types of encoded information.

- **Factual.** This refers to statements such has *The Matrix was directed by the Wachowskis* or *Melbourne is located in Australia* that describe attributes of an entity;

- **Categorical.** It is mainly used to state something about the subject of an

[2] `https://googleblog.blogspot.it/2012/05/introducing-knowledge-graph-things-not.html`

[3] `http://linkeddata.org`

entity. In this direction, the categories of Wikipedia pages are an excellent example. Categories can be used to cluster entities and are often organized hierarchically thus making possible to define them in a more generic or specific way;

- **Ontological.** This is a more restrictive and formal way to classify entities via a hierarchical structure of classes. Differently from categories, sub-classes and super-classes are connected through IS-A (transitive) relations.

If we take a look at the following RDF triples[4] from the DBpedia knowledge graph

```
dbr:The_Matrix dbo:director dbr:The_Wachowski_Brothers.
dbr:The_Matrix dct:subject dbc:Dystopian_films.
dbr:The_Matrix rdf:type dbo:Film.
```

we may see that each of them represents one of the above mentioned types of data. In the first triple we state a fact about the movie *The Matrix* (represented by the corresponding URI `dbr:The_Matrix`) saying that is has been directed (`dbo:director`) by the *Wachowski Brothers* (`dbr:The_Wachowski_Brothers`). The second triple encodes categorical information through the predicate `dct:subject` about the same movie. In particular, here we say that it belongs to the category of dystopian films (`dbc:Dystopian_films`). Finally, with the last triple, we classify *The Matrix* as a Film (`dbo:Film`) thanks to the predicate `rdf:type`.

7.3.2 Formal Model

Factorization models have been proven to be among of the best performing approaches as collaborative filtering methods to build a recommender system [217]. This is due to the high prediction accuracy and the subtle modeling of user-item interactions which let these models operate efficiently even in very sparse settings (compared to other classical collaborative predictive models). Among all the different factorization models, factorization machines propose a unified general model

[4]For the sake of conciseness we will use the CURIE syntax in which URIs are abbreviated using their namespaces. In this study we refer to namespaces as available at `http://prefix.cc`.

to represent most of them. Here we report the definition and results related to a factorization model of order 2 for a recommendation problem involving only implicit ratings. Nevertheless, everything can be easily extended to a more expressive representation by taking into account, e.g., demographic and social information [8], multi-criteria [6], and even relations between contexts [301]. For each user $u \in U$ and each item $i \in I$ we build a binary vector $\mathbf{x^{ui}} \in \mathbb{R}^{1\times n}$, with $n = |U| + |I|$, representing the interaction between u and i in the original user-item rating matrix. In this modeling, $\mathbf{x^{ui}}$ contains only two 1 values corresponding to u and i while all the other values are set to 0 (see Fig. 7.1). We then denote with $\mathbf{X} \in \mathbb{R}^{n\times m}$ the matrix containing as rows all possible $\mathbf{x^{ui}}$ we can build starting from the original user-item rating matrix as shown in Fig. 7.1.

	User					Item					
	U_1	U_2	U_3	U_4	...	I_1	I_2	I_3	I_4	I_5	...
x^1	1	0	0	0	...	1	0	0	0	0	...
x^2	1	0	0	0	...	0	1	0	0	0	...
x^3	1	0	0	0	...	0	0	1	0	0	...
x^4	0	1	0	0	...	0	0	1	0	0	...
x^5	0	1	0	0	...	0	0	0	1	0	...
x^6	0	0	1	0	...	1	0	0	0	0	...
x^7	0	0	1	0	...	0	0	1	0	0	...

Figure 7.1: A visual representation of $\mathbf{X}$ for sparse real valued vectors $\mathbf{x^{ui}}$.

The factorization machine (FM) for each vector $\mathbf{x}$ can be defined as:

$$\hat{y}(\mathbf{x^{ui}}) = w_0 + \sum_{j=1}^{n} w_j \cdot x_j + \sum_{j=1}^{n} \sum_{j'=j+1}^{n} x_j \cdot x_{j'} \cdot \sum_{f=1}^{k} v_{(j,f)} \cdot v_{(j',f)} \tag{7.1}$$

where the parameters to be learned are: w_0 representing the global bias; w_j giving the importance to every single x_j; the pair $v_{(j,f)}$ and $v_{(j',f)}$ in $\sum_{f=1}^{k} v_{(j,f)} \cdot v_{(j',f)}$ measuring the strength of the interaction between each pair of variables x_j and $x_{j'}$. The number of latent factors is represented by k. This value is usually selected at design time when implementing the FM.

In order to make the recommendation results computed by `kaHFM` semantically interpretable, we want to inject the knowledge encoded within a knowledge-graph in a Factorization Machine. Given a set of features retrieved from a KG [189] we first bind them to the latent factors and then, since we address a Top-N recommendation problem, we train the model by using a Bayesian Personalized Ranking (BPR) criterion that takes into account entities within the original knowledge graph.

In [191], the authors originally proposed to encode a Linked Data knowledge graph in a vector space model to develop a CB recommender system. Given a set of items $I = \{i_1, i_2, \ldots, i_N\}$ in a catalog and their associated triples $\langle i, \rho, \omega \rangle$ in a knowledge graph $\mathcal{KG}$ we may build the set of all possible features as $F = \{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathcal{KG} \text{ with } i \in I\}$. Each item can be then represented as a vector of weights $\mathbf{i} = [v_{(i,1)}, \ldots, v_{(i,\langle \rho,\omega \rangle)}, \ldots, v_{(i,|F|)}]$ where $v_{(i,\langle \rho,\omega \rangle)}$ is computed as the normalized TF-IDF value for $\langle \rho, \omega \rangle$:

$$
\begin{aligned}
v_{(i,\langle \rho,\omega \rangle)} \quad = \quad & \underbrace{\frac{|\{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathcal{KG}\}|}{\sqrt{\sum\limits_{\langle \rho,\omega \rangle \in F} |\{\langle \rho, \omega \rangle \mid \langle i, \rho, \omega \rangle \in \mathcal{KG}\}|^2}}}_{TF^{\mathcal{KG}}} \cdot \\
\cdot \quad & \underbrace{\log \frac{|I|}{|\{j \mid \langle j, \rho, \omega \rangle \in \mathcal{KG} \text{ and } j \in I\}|}}_{IDF^{\mathcal{KG}}}
\end{aligned} \tag{7.2}
$$

Analogously, when we have a set U of users, we may represent them using the features describing the items they enjoyed in the past. In the following, when no confusion arises, we use f to denote a feature $\langle \rho, \omega \rangle\ inF$. Given a user u, if we denote with I^u the set of the items enjoyed by u and we have $\mathbf{u} = [v_{(u,1)}, \ldots, v_{(u,f)} \ldots, v_{(u,|F|)}]$ with

$$
v_{(u,f)} = \frac{\sum\limits_{i \in I^u} v_{(i,f)}}{|\{i \mid i \in I^u \text{ and } v_{(i,f)} \neq 0\}|}
$$

Given the vectors $\mathbf{u}_j$, with $j \in [1 \ldots |U|]$, and $\mathbf{i}_{j'}$, with $j' \in [1 \ldots |I|]$, we build the matrix $\mathbf{V} \in \mathbb{R}^{n \times |F|}$ (see Fig. 7.2) where the first $|U|$ rows have a one to one mapping with $\mathbf{u}_j$ while the last ones correspond to $\mathbf{i}_{j'}$. If we go back to Equation (7.1) we may see that, for each $\mathbf{x}$, the term $\sum_{j=1}^{n} \sum_{j'=j+1}^{n} x_j \cdot x_{j'} \cdot \sum_{f=1}^{k} v_{(j,f)} \cdot v_{(j',f)}$ is not zero only once, i.e., when both x_j and $x_{j'}$ are equal to 1. In the matrix depicted in Fig. 7.1, this happens when there is an interaction between a user and an item. Moreover, the summation $\sum_{f=1}^{k} v_{(j,f)} \cdot v_{(j',f)}$ represents the dot product between two vectors $\mathbf{v}_j$

	dbc:Space_adventure_films	dbc:Films_set_in_the_future	dbc:American_science_fiction_action_films	dbc:1980s_science_fiction_films	dbc:Paramount_Pictures_films	dbc:Midlife_crisis_films	dbc:American_sequel_films	
v_1	**0**	0.88	0.81	0.7	0	0.60	0.53	...
v_2	1.3	1.12	0.91	0.84	0.65	0.59	0.58	...
v_3	0.5	0	0.71	0	0.28	0.35	0	...
v_4	0	0	0.31	0	0	0	0.6	...
v_5	0	0	0	0	0.18	0	0	...
v_6	0	0.12	0.22	0	0	0	0	...
v_7	1.23	1.03	0.89	0.85	0.56	0.3	0.61	...

Figure 7.2: Example of real valued feature vectors for different items v_j.

and $\mathbf{v}_{j'}$ with a size equal to k. Hence, $\mathbf{v}_j$ represents a latent representation of a user, $\mathbf{v}_{j'}$ that of an item within the same latent space, and their interaction is evaluated through their dot product.

In order to inject the knowledge coming from $\mathscr{KG}$ into `kaHFM`, first of all, we keep Equation (7.1) and we set $k = |F|$. In other words, we impose a number of latent factors equal to the number of features describing all the items in our catalog. We want to stress here that our aim is not representing each feature through a latent vector, but to associate each factor to an explicit feature, obtaining latent vectors that are composed by explicit semantic features. Hence, we initialize the parameters $\mathbf{v}_j$ and $\mathbf{v}_{j'}$ with their corresponding rows from $\mathbf{V}$ which in turn represent respectively $\mathbf{u}_j$ and $\mathbf{i}_{j'}$. In this way, we try to identify each latent factor with a corresponding explicit feature. The intuition is that after the training phase, the resulting matrix $\hat{\mathbf{V}}$ still refers to the original features but contains better values for $v_{(j,f)}$ and $v_{(j',f)}$ that also take into account the latent interactions between users, items and features. It is noteworthy that after the training phase $\mathbf{u}_j$ and $\mathbf{i}_{j'}$ (corresponding to $v_{(j,f)}$ and $v_{(j',f)}$ in $\mathbf{V}$) contain non-zero values also for features that are not originally in the description of the user u or of the item i.

In Table 7.1 and Table 7.2 we show examples for values after the training (in the column kaHFM) together with the original TF-IDF ones computed for two movies from the Yahoo! Movies[5] dataset.

kaHFM	TF-IDF	Predicate	Object
1.3669	0.2584	dct:subject	dbc:Space_adventure_films
1.1252	0.2730	dct:subject	dbc:Films_set_in_the_future
0.9133	0.2355	dct:subject	dbc:American_science_fiction_action_films
0.8485	0.3190	dct:subject	dbc:1980s_science_fiction_films
0.6529	0.1549	dct:subject	dbc:Paramount_Pictures_films
0.5989	0.3468	dct:subject	dbc:Midlife_crisis_films
0.5940	0.1797	dct:subject	dbc:American_sequel_films
0.5862	0.2661	dct:subject	dbc:Film_scores_by_James_Horner
0.5634	0.2502	dct:subject	dbc:Films_shot_in_San_Francisco
0.5583	0.1999	dct:subject	dbc:1980s_action_thriller_films

Table 7.1: Top-10 features computed by kaHFM for the movie "Star Trek II - The Wrath of Khan".

kaHFM	TF-IDF	Predicate	Object
1.2434	0.2858	dct:subject	dbc:Space_adventure_films
1.0355	0.3020	dct:subject	dbc:Films_set_in_the_future
0.8956	0.2605	dct:subject	dbc:American_science_fiction_action_films
0.8951	0.3451	dct:subject	dbc:Android_(robot)_films
0.7338	0.3105	dct:subject	dbc:Time_travel_films
0.6665	0.2701	dct:subject	dbc:Film_scores_by_Jerry_Goldsmith
0.6581	0.2205	dct:subject	dbc:1990s_action_films
0.6561	0.2279	dct:subject	dbc:1990s_science_fiction_films
0.6118	0.1988	dct:subject	dbc:American_sequel_films
0.5649	0.1713	dct:subject	dbc:Paramount_Pictures_films

Table 7.2: Top-10 features computed by kaHFM for the movie "Star Trek - First Contact".

7.3.3 Optimization

Factorization machines can be easily trained to reduce the prediction error via gradient descent methods, alternating least-squares (ALS) and MCMC. Since we formulated our problem as a *top-N* recommendation task, kaHFM can be trained using a learning to rank approach like Bayesian Personalized Ranking Criterion (BPR)

[5] http://research.yahoo.com/Academic_Relations

[218]. The BPR criterion is optimized using a stochastic gradient descent algorithm on a set D_S of triples (u,i,j), with $i \in I^u$ and $j \notin I^u$, selected through a random sampling from a uniform distribution. The BPR optimization criterion can thus be formulated as:

$$\begin{aligned}\texttt{BPR-OPT} &= \sum_{(u,i,j)\in D_s} ln\sigma(\hat{x}_{uij}) - \lambda_\Theta \|\Theta\|^2 \\ &= \sum_{(u,i,j)\in D_s} ln\sigma(\hat{y}(\mathbf{x^{ui}}) - \hat{y}(\mathbf{x^{uj}})) - \lambda_\Theta \|\Theta\|^2 \end{aligned} \tag{7.3}$$

In this formulation, $\sigma(\cdot)$ is a sigmoid function, and the update step is defined as:

$$\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial\Theta}\hat{x}_{uij} + \lambda\Theta \right) \tag{7.4}$$

where α is the chosen learning rate. Since we are in an implicit feedback setting, we may assume that there is only an instance for the pair user-item. Hence, in our model we can derive $\hat{x}_{uij}$ as:

$$\begin{aligned}\hat{x}_{uij} &= \hat{y}(\mathbf{x^{ui}}) - \hat{y}(\mathbf{x^{uj}}) = w_i x_i - w_j x_j + \\ &+ \sum_{f\in F} x_u x_i v_{(u,f)} v_{(i,f)} - x_u x_j v_{(u,f)} v_{(j,f)} \end{aligned} \tag{7.5}$$

For `kaHFM`, this computation can be performed in an efficient way computing the partial derivatives (to update the factorized parameters) for the only active entities involved in the transactions, w_i, w_i, v_u, v_i, and v_j:

$$\frac{\partial}{\partial\Theta}\hat{x}_{uij} = \begin{cases} 1, & \text{if } \theta = w_i, \\ -1, & \text{if } \theta = w_j, \\ v_{(u,f)}, & \text{if } \theta = v_{(i,f)}, \\ -v_{(u,f)}, & \text{if } \theta = v_{(j,f)}, \\ (v_{(i,f)} - v_{(j,f)}), & \text{if } \theta = v_{(u,f)}, \\ 0, & \text{otherwise} \end{cases} \tag{7.6}$$

Using Equation (7.6) in Equation (7.4) the model parameters can be iteratively updated to maximize the BPR-OPT criterion. The algorithm updates sequentially each sampled triple and continues the training until it reaches the provided number of iterations.

7.3.4 Personalized Recommendation

Once the training phase returns the optimal model parameters, the item recommendation step can take place. We extract the items vectors $\mathbf{v}_j$ from the matrix $\mathbf{V}$, with the associated optimal values and we use them to implement an Item-kNN recommendation approach. We measure similarities between each pair of items i and j by evaluating the cosine similarity of their corresponding vectors in $\mathbf{V}$:

$$cs(i,j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\parallel \mathbf{v}_i \parallel \cdot \parallel \mathbf{v}_j \parallel}$$

Given a set of neighbors for the item i, denoted as N^i, such that $i \notin I^u$ and a user u we may predict the score assigned by u to i as

$$score(u,i) = \frac{\sum_{j \in N^i \cap I^u} cs(i,j)}{\sum_{j \in N^i} cs(i,j)} \tag{7.7}$$

The proposed approach let us keep the explicit meaning of the "latent" factors computed via a factorization machine thus making possible an interpretation of the recommended results. In the following we propose an automated offline procedure able to assess that the semantics of the features represented in $\mathbf{V}$ is preserved after the training phase. We refer to the values computed by the proposed procedure as *Semantic Accuracy*. A different but related aspect is that of evaluating if kaHFM really assigns a higher value to meaningful features. We refer to this behavior as *Robustness*. Interestingly, both *Semantic Accuracy* and *Robustness* can be evaluated in an offline setting.

7.3.5 Semantic Accuracy

The main idea behind Semantic Accuracy is to evaluate, given an item i, how well kaHFM is able to return its original features available in the returned top-K list $\mathbf{v}_i$. In other words, given the set of features of i represented by $F^i = \{f_1^i, \dots, f_m^i, \dots f_M^i\}$, with $F^i \subseteq F$, we check if the values in $\mathbf{v}_i$ corresponding to $f_{m,i} \in F^i$ are higher than those corresponding to $f \notin F^i$. For the set of M features initially describing i we see how many of them appear in the set $top(\mathbf{v}_i, M)$ representing the top-M features in $\mathbf{v}_i$. We then normalize this number by the size of F^i and average on all the items within the catalog I.

$$\texttt{Semantic Accuracy (SA@}M\texttt{)} = \frac{\sum_{i \in I} \frac{|top(\mathbf{v}_i, M) \cap F^i|}{|F^i|}}{|I|}$$

In many practical scenarios we may have $|F| \gg M$. Hence, we might also be interested in measuring the accuracy for different sizes of the top list. Since items could be described with a different number of features, the size of the top list could be a function of the original size of the item description. Thus, we measured SA@nM with $n \in \{1,2,3,4,5,\dots\}$ and evaluate the number of features in F^i available in the top-$n \cdot M$ elements of $\mathbf{v}_i$.

$$\texttt{SA@}nM = \frac{\sum_{i \in I} \frac{|top(\mathbf{v}_i, n \cdot M) \cap F^i|}{|F^i|}}{|I|}$$

7.3.6 Robustness

Although SA@nM may result very useful to understand if kaHFM assigns weights according to the original description of item i, we still do not know if a high value in $\mathbf{v}_i$ really means that the corresponding feature is important to define i. In other words, are we sure that kaHFM promotes meaningful features for i?

In order to provide a way to measure such “meaningfulness” for a given feature, we suppose, for a moment, that a particular feature $\langle \rho, \omega \rangle$ is useful to describe an item i but the corresponding triple $\langle i, \rho, \omega \rangle$ is not represented in the knowledge graph. In case kaHFM was robust in generating weights for unknown features, it should discover the importance of that feature and modify its value to make it enter the Top-K features in $\mathbf{v}_i$.

Starting from this observation, the idea to measure robustness is then to "forget" a triple involving i and check if `kaHFM` can generate it.

In order to implement such process we may proceed by following these steps:

- we train `kaHFM` thus obtaining optimal values v_i for all the features in F^i;
- the feature $f^i_{MAX} \in F^i$ with the highest value in v_i is identified;
- we train the model again initializing $f^i_{MAX} = 0$ and we compute v'_i.

After the above steps, if $f^i_{MAX} \in top(v'_i, M)$ then we can say that `kaHFM` shows a high robustness in identifying important features.

Given a catalog I, we may then define the *Robustness for 1 removed feature @M* (`1-Rob@M`) as the number of items for which $f^i_{MAX} \in top(v'_i, M)$ divided by the size of I.

$$\texttt{1-Rob@M} = \frac{\sum_{i \in I} |\{i \mid f^i_{MAX} \in top(v'_i, M)\}|}{|I|}$$

Similarly to `SA@`nM, we may define `1-Rob@nM`.

7.4 Experimental Evaluation

7.4.1 Experimental Setup

Datasets. In order to provide an answer to RQ1, we evaluated the performance of our method on six datasets belonging to three different domains (Music, Books and Movies). The `Last.fm` dataset [53] corresponds to user-artist plays on Last.fm online music system released during HETRec 2011[6] Workshop. It contains social networking, tagging, and music artists listening information from a set of 2K users. `LibraryThing` represents books' ratings collected in the LibraryThing website[7]

[6] `http://ir.ii.uam.es/hetrec2011/`

[7] `https://www.librarything.com/`

community. It contains social networking, tagging and rating information on a [1..10] scale. Yahoo!Movies (Yahoo! Webscope dataset ydata-ymovies-user-movie-ratings-content-v1_0)[8] contains movies ratings generated on Yahoo! Movies up to November 2003. It provides content, demographic and ratings information on a [1..5] scale, and mappings to MovieLens and EachMovie datasets. Facebook Movies, Facebook Music and Facebook Books datasets have been released for the Linked Open Data challenge co-located with ESWC 2015[9], and they refer to movies, music and books domains respectively. Only implicit feedback is available for these datasets, but for each item a link to DBpedia is provided. In order to map items in Last.fm and LibraryThing to DBpedia resources, we exploited a freely available mapping[10]. For the remaining one (Yahoo!Movies), we extracted all the updated items-features mappings (Yahoo!Movies, LibraryThing, Last.fm, Facebook Movies, Facebook Music and Facebook Books) which are publicly available[11].

Datasets statistics are shown in Table 7.3.

Dataset	#Users	#Items	#Transactions	#Features	Sparsity
Yahoo! Movies	4000	2,626	69,846	988,734	99.34%
LibraryThing	7223	11,695	410,210	183,182	99.51%
Last FM	1375	8,289	60,701	434,817	99.47%
Facebook Music	52068	5,749	1,374,994	345,452	99.54%
Facebook Movies	32143	3,901	689,561	180,573	99.45%
Facebook Books	1398	2,933	18,978	111,401	99.53%

Table 7.3: Datasets statistics.

Evaluation Protocol and Experimental Setting. "All Unrated Items" [253] protocol has been chosen to compare different algorithms. In All Unrated Items, for each user, all the items that have not yet been rated by the user all over the catalog are considered. We split the dataset using Hold-Out 80-20 retaining for every user the 80% of their ratings in the training set and the remaining 20% in the test set.

[8] http://research.yahoo.com/Academic_Relations

[9] https://2015.eswc-conferences.org/program/semwebeval.html

[10] https://github.com/sisinflab/LODrecsys-datasets

[11] https://github.com/sisinflab/LinkedDatasets/

Moreover, a temporal split has been performed [112] whenever timestamps associated to every transaction is available. We tested our approach against the most related content-based and collaborative algorithms in terms of Accuracy, Diversity and Novelty [11]. We compared `kaHFM`[12] w.r.t. a canonical 2 degree Factorization Machine (users and items are intended as features of the original formulation) by optimizing the recommendation list ranking via BPR (BPR-FM). In order to preserve the fairness of the comparison, we used the same parameters adopted for `kaHFM` and the same number of hidden factors (see the "Selected" column in Table 7.5). Moreover, since we use items similarity in the last step of our approach (see Equation (7.7)), we compared `kaHFM` against an *Attribute Based Item-kNN* (ABItem-kNN) algorithm, where each item is represented as a vector of weights, computed through a TF-IDF model. In this model, the attributes are computed via Equation (7.2). For the sake of completeness we also compared `kaHFM` against a pure Item-kNN, that is an item-based implementation of the k-nearest neighbors algorithm. It finds the k-nearest item neighbors based on Cosine Similarity. Items in the neighborhood are then used to predict a score for each user-item pair. Regarding BPR parameters, *learning rate*, *bias regularization*, *user regularization*, *positive item regularization*, and *negative item regularization* have been set respectively to 0.05, 0, 0.0025, 0.0025 and 0.00025 while a sampler "without replacement" has been adopted in order to sample the triples as suggested by authors[218]. For the sake of reproducibility, the BPR parameters are chosen from `mymedialite` implementation. Moreover, the same parameters are used in all the algorithms that make use of BPR to avoid affecting the results, and to guarantee a fair comparison. We also compared `kaHFM` against the corresponding User-based nearest neighbor scheme, and Most-Popular, a simple baseline that shows high performance in specific scenarios [69]. In our context, we considered mandatory to also compare against a pure knowledge-graph content-based baseline based on Vector Space Model (VSM) [191].

[12] Implementation available at `https://github.com/sisinflab/HybridFactorizationMachines/`

7.4.2 Features extraction

The feature extraction is one of the most sensitive steps in our approach. A wrong feature selection may result in noisy data, or in the lack of some important features. This preprocessing was basically divided in three steps: (i) "**Extraction**", in which we retrieve data from the DBpedia knowledge graph, (ii) "**Selection**" where only features involved in the specific experiment are selected, and (iii) "**Filtering**" in which uninformative features are removed [189].

Extraction. Thanks to the publicly available mappings, all the items from the datasets represented in Table 7.3 come with a `DBpedia` link. Exploiting this reference, we retrieved all the $\langle \rho, \omega \rangle$ pairs. Some noisy features (based on the following predicates) have been excluded already in this early extraction. In particular we removed: `owl:sameAs`, `dbo:thumbnail`, `prov:wasDerivedFrom`, `foaf:depiction`, `foaf:isPrimaryTopicOf`. Behind this choice, the main reason is that they give us only information about the nature of the entity in the specific knowledge base (e.g., the links between `DBpedia` and `WikiPedia` pages) or are linked to multimedia data or even external datasets. After this cleaning step, all the features have been indexed, saved separately and associated to the *id* of the item.

Selection. We performed our experiments with three different settings to analyze the impact of the different kind of features in the recommendation accuracy and diversity. The features have been chosen as they are present in all the different domains and because of their factual, categorical or ontological meaning:

- **Categorical Setting (CS)**: We selected only the features containing the property `dcterms:subject`.
- **Ontological Setting (OS)**: In this case the only features we considered are: `rdf:type`, `dbo:genre`, `dcterms:subject`.
- **Factual Setting (FS)**: We considered all the features but those involving the properties selected in OS.

Dataset	Threshold
Yahoo! Movies	99.62
LibraryThing	99.91
Last FM	99.88
Facebook Music	99.83
Facebook Movies	99.74
Facebook Books	99.66

Table 7.4: Threshold values adopted to filter out irrelevant properties for each dataset.

Filtering. This last step corresponds to the removal of irrelevant features, that bring little value to the recommendation task, but, at the same time, pose scalability issues. The pre-processing phase has been done following [189], and [208] with a unique threshold. The corresponding thresholds (*tm* [189], and *p* [208] for missing values) for each dataset are represented in Table 7.4.

We discarded features for which we had more than *tm* (or, equivalentely *p*) missing values. For a fair comparison the features used for the baseline Attribute-based Item kNN (ABItem-kNN) are the same used in our approach and the number of latent factors for FMs has been set equal to the number of features involved in the specific setting. The characteristics of each datasets (varying the setting) in terms of considered features are reported in Table 7.5.

	Categorical Setting		**Ontological Setting**		**Factual Setting**	
Datasets	Total	Selected	Total	Selected	Total	Selected
`Yahoo!Movies`	26155	747	38699	1240	950035	3186
`LibraryThing`	9443	1169	14585	1934	168597	5826
`Last.fm`	16422	1315	30734	3032	404083	9413
`Facebook Music`	15016	1057	27988	2531	317464	7881
`Facebook Movies`	8843	1103	13828	1848	166745	5427
`Facebook Books`	6231	263	9881	592	101520	1315

Table 7.5: Considered features in the different settings

7.4.3 Accuracy, Diversity and Novelty with kaHFM

In order to evaluate our approach, we measured accuracy, novelty and aggregate diversity metrics. Accuracy metrics were measured through Precision@N (*Prec@N*) and Normalized Discounted Cumulative Gain (*nDCG@N*). This latter measures the usefulness of an item based on its position in the recommendation list. Hence, it has been computed only for datasets that have explicit ratings (i.e., LibraryThing and Yahoo!Movies). Since the recommended results may vary in length depending on the user, cumulative gain at each position is normalized across users. *EPC* (Expected Popularity Complement) is used to measure novelty, or more precisely the ability of the algorithm to select items that belong to the long tail. Finally, diversity has been measured through *Catalog Coverage* (aggregate diversity in *top-N* list), *Gini Index* and *Shannon entropy*. In particular, *Catalog Coverage* denotes the ability of a system in selecting as many elements as possible from the whole catalog while *Gini index* (Gini) and *Shannon entropy* are used to measure the distributional inequality with different approaches. Both accuracy and novelty metrics have been computed by averaging their values per-user. To compute those metrics we used the implementation provided by RankSys[13] framework. The evaluation has been performed considering Top-10 ([57, 69, 245]) recommendations for all the datasets. When a rating score was available, a *Threshold-based relevant items* condition [52] was adopted in order to take into account only relevant items. In particular, a relevance threshold of 4/5 and 8/10 has been set for Yahoo!Movies and LibraryThing respectively.

Tables 7.6, and 7.7 show the results of our experiments regarding accuracy and diversity.

In all the tables we highlight in **bold** the best result while we underline the second one. Statistically significant results are denoted with a * mark. We used a Student's paired t-test with a 0.05 level.

LibraryThing experiments show that our approach outperforms the competing algorithm for all the considered metrics. It is worth to notice that the *nDCG@N*

[13] http://ranksys.org/

	Facebook Movies					Facebook Music					Facebook Books				
Categorical Setting (CS)	P@10	EPC	AD@10	Gini	SE	P@10	EPC	AD@10	Gini	SE	P@10	EPC	AD@10	Gini	SE
ABItem-kNN	0.0173*	0.0196*	**3566**	0.2217	9.5537	0.0200*	0.0195*	**4961**	**0.1800**	**10.2039**	0.0060*	0.0056*	1343	0.1498	9.0882
BPR-FM	0.0158*	0.0149*	179	0.0043	4.5639	0.0138*	0.0107*	544	0.0061	5.4518	0.0036*	0.0034*	78	0.0062	4.6815
MostPopular	0.0118*	0.0099*	27	0.0029	3.8543	0.0146*	0.0089*	30	0.0020	3.8628	0.0032*	0.0030*	17	0.0034	3.6193
ItemKnn	0.0262*	0.0270*	2554	0.0963	8.7266	0.0279*	0.0269*	3916	0.0960	9.2819	0.0041*	0.0042*	1437	0.1759	9.2526
UserKnn	0.0168*	0.0157*	466	0.0104	5.7206	0.0130*	0.0109*	961	0.0157	6.7938	0.0101*	0.0095*	269	0.0150	5.8499
VSM	0.0185*	0.0205*	3326	0.1769	9.5856	0.0289*	0.0325*	4582	0.1395	9.6626	0.0104*	0.0112*	1833	0.2631	9.8735
kaHFM	**0.0296**	**0.0324**	3560	**0.2493**	**10.1243**	**0.0338**	**0.0353**	4373	0.1166	9.5861	**0.0129**	**0.0136**	**1905**	**0.3128**	**10.1855**
Ontological Setting (OS)	P@10	EPC	AD@10	Gini	SE	P@10	EPC	AD@10	Gini	SE	P@10	EPC	AD@10	Gini	SE
ABItem-kNN	0.0172	0.0188	**3646**	**0.2589**	9.9645	**0.0295**	**0.0314**	5152	0.2021	10.2691	0.0109*	0.0113*	2004	0.2954	10.0712
BPR-FM	0.0155*	0.0128*	144	0.0040	4.4288	0.0083	0.0065	207	0.0036	4.8614	0.0037*	0.0034*	85	0.0062	4.6887
MostPopular	0.0118*	0.0099*	27	0.0029	3.8543	0.0146	0.0089	30	0.0020	3.8628	0.0032*	0.0030*	17	0.0034	3.6193
ItemKnn	0.0263*	0.0271*	2557	0.0963	8.7270	0.0280	0.0270	3919	0.0960	9.2813	0.0041*	0.0042*	1437	0.1759	9.2526
UserKnn	0.0168*	0.0157*	466	0.0104	5.7207	0.0130	0.0109	961	0.0157	6.7938	0.0101*	0.0095*	269	0.0150	5.8493
VSM	0.0181*	0.0198*	3255	0.1765	9.5424	0.0257	0.0274	4231	0.1180	9.3889	0.0072*	0.0085*	1582	0.1691	9.1551
kaHFM	**0.0273**	**0.0300**	3631	0.2500	**10.1188**	0.0276	0.0288	**5391**	**0.2537**	**10.5843**	**0.0148**	**0.0162**	**2099**	**0.3183**	**10.1984**
Factual Setting (FS)	P@10	EPC	AD@10	Gini	SE	P@10	EPC	AD@10	Gini	SE	P@10	EPC	AD@10	Gini	SE
ABItem-kNN	0.0234	0.0275	3605	0.2003	9.5874	0.0246	0.0258	5232	0.2336	10.4700	0.0147*	0.0163*	2098	0.3105	10.1480
BPR-FM	0.0157	0.0133	110	0.0039	4.3774	0.0119	0.0091	357	0.0049	5.2379	0.0041*	0.0037*	90	0.0067	4.7874
MostPopular	0.0123	0.0102	26	0.0029	3.8531	0.0114	0.0070	31	0.0020	3.8674	0.0033*	0.0031*	17	0.0034	3.6195
ItemKnn	**0.0273**	**0.0283**	2617	0.0983	8.7616	0.0289	0.0278	3993	0.0982	9.3085	0.0041*	0.0040*	1656	0.2117	9.4983
UserKnn	0.0176	0.0165	470	0.0106	5.7393	0.0136	0.0114	987	0.0159	6.8209	0.0109*	0.0101*	271	0.0157	5.9059
VSM	0.0219	0.0245	2812	0.0909	8.5810	**0.0348**	**0.0366**	3846	0.0925	9.1179	0.0126*	0.0140*	1862	0.2641	9.9322
kaHFM	0.0240	0.0268	**3619**	**0.2434**	**10.1562**	0.0313	0.0336	**5350**	**0.2491**	**10.4870**	**0.0179**	**0.0189**	**2211**	**0.3523**	**10.3441**

Table 7.6: Accuracy, Diversity and Novelty results for Facebook Movies, Facebook Music and Facebook Books

	LibraryThing						Yahoo!Movies						Last.fm				
Categorical Set.	P@10	nDCG@10	EPC	AD@10	Gini	SE	P@10	nDCG@10	EPC	AD@10	Gini	SE	P@10	EPC	AD@10	Gini	SE
ABItem-kNN	0.0408*	0.0460*	0.0424*	6335	0.1607	11.1013	0.0421*	0.1174*	0.0528*	**2447**	**0.3640**	**10.0302**	0.0223	0.0210	3642	0.1823	10.6194
BPR-FM	0.0151*	0.0162*	0.0138*	527	0.0029	5.2783	0.0189*	0.0344*	0.0184*	123	0.0056	4.3270	0.0314	0.0327	158	0.0022	4.6219
MostPopular	0.0056*	0.0058*	0.0051*	34	0.0009	3.8302	0.0154*	0.0271*	0.0149*	48	0.0043	3.9046	0.0252	0.0233	35	0.0012	3.7101
ItemKnn	0.0425*	0.0590*	0.0477*	5939	0.1396	10.8123	0.0203*	0.0427*	0.0193*	1442	0.1486	8.9548	**0.0449**	0.0405	2318	0.0947	9.8270
UserKnn	0.0213*	0.0346*	0.0226*	1330	0.0103	6.7102	0.0231*	0.0474*	0.0232*	729	0.0336	6.6712	0.0394	**0.0443**	1127	0.0248	7.6857
VSM	0.0367*	0.0472*	0.0393*	7431	0.2106	11.4202	0.0385*	0.1129*	0.0496*	2320	0.2893	9.7604	0.0025	0.0023	4582	0.0967	9.6626
kaHFM	**0.0639**	**0.0913**	**0.0726**	**9367**	**0.3139**	**12.1071**	**0.0524**	**0.1399**	**0.0613**	2433	0.3406	9.9831	0.0354	0.0364	**4732**	**0.2976**	**11.5368**
Ontological Set.	P@10	nDCG@10	EPC	AD@10	Gini	SE	P@10	nDCG@10	EPC	AD@10	Gini	SE	P@10	EPC	AD@10	Gini	SE
ABItem-kNN	0.0446*	0.0539*	0.0485*	7669	0.2139	11.4349	0.0427*	0.1223*	0.0545*	**2461**	**0.3634**	**10.0776**	0.0297	0.0308	4278	0.2507	11.1528
BPR-FM	0.0121*	0.0126*	0.0108*	396	0.0023	5.0055	0.0199*	0.0356*	0.0195*	111	0.0053	4.2386	0.0287	0.0304	155	0.0022	4.5842
MostPopular	0.0056*	0.0058*	0.0051*	34	0.0009	3.8302	0.0154*	0.0271*	0.0149*	48	0.0043	3.9046	0.0252	0.0233	35	0.0012	3.7101
ItemKnn	0.0425*	0.0591*	0.0477*	5939	0.1396	10.8125	0.0203*	0.0427*	0.0193*	1442	0.1486	8.9548	0.0449	0.0405	2318	0.0947	9.8270
UserKnn	0.0213*	0.0346*	0.0226*	1330	0.0103	6.7102	0.0232*	0.0474*	0.0232*	729	0.0336	6.6711	**0.0526**	**0.0577**	823	0.0163	7.2512
VSM	0.0367*	0.0472*	0.0393*	7431	0.2106	11.4202	0.0349*	0.1083*	0.0450*	2216	0.2706	9.7345	0.0025	0.0026	4231	0.0819	9.3889
kaHFM	**0.0635**	**0.0912**	**0.0728**	**9083**	**0.3081**	**12.0563**	**0.0521**	**0.1380**	**0.0608**	2444	0.3442	10.0086	0.0371	0.0381	**4853**	**0.3200**	**11.6318**
Factual Set.	P@10	nDCG@10	EPC	AD@10	Gini	SE	P@10	nDCG@10	EPC	AD@10	Gini	SE	P@10	EPC	AD@10	Gini	SE
ABItem-kNN	0.0488*	0.0596*	0.0513*	7419	0.1968	11.2464	0.0619	**0.1764**	**0.0777**	**2433**	0.3177	9.7847	0.0376	0.0362	4179	0.2379	11.0850
BPR-FM	0.0087*	0.0087*	0.0080*	184	0.0015	4.5400	0.0177	0.0305	0.0171	116	0.0054	4.2903	0.0233	0.0253	1815	0.0347	6.4129
MostPopular	0.0056*	0.0058*	0.0051*	34	0.0009	3.8301	0.0154	0.0271	0.0149	48	0.0043	3.9046	0.0252	0.0233	35	0.0012	3.7103
ItemKnn	0.0436*	0.0615*	0.0491*	6162	0.1471	10.9008	0.0203	0.0427	0.0193	1442	0.1486	8.9548	0.0426	0.0372	2351	0.0955	9.8092
UserKnn	0.0217*	0.0349*	0.0228*	1399	0.0108	6.7459	0.0232	0.0474	0.0232	729	0.0336	6.6711	0.0403	**0.0451**	1163	0.0256	7.7004
VSM	0.0456*	0.0575*	0.0496*	6860	0.1928	11.3774	**0.0627**	0.1725	0.0752	2203	0.2251	9.1183	**0.0430**	0.0419	3325	0.1590	10.4706
kaHFM	**0.0627**	**0.0906**	**0.0713**	**9089**	**0.3134**	**12.0663**	0.0564	0.1434	0.0639	2394	**0.3511**	**10.1138**	0.0339	0.0352	**4788**	**0.3139**	**11.6257**

Table 7.7: Accuracy, Diversity and Novelty results for LibraryThing, Yahoo!Movies and Last.fm

is almost doubled when using kaHFM. The worst results are achieved by Most-Popular, followed by BPR-FM and this happens in almost all the experiments. This is probably due to the high number of latent factors that does not make the method perform efficiently. Yahoo!Movies experiments show that in Categorical and Ontological settings our method is the most accurate, while the diversity performance of kaHFM w.r.t. ABItem-kNN are quite similar. In the Yahoo!Movies mapping a strong popularity bias is present and it is interesting to notice that this affects only the Factual setting leading our approach to be less precise than ABItem-kNN while our method proposes a bit more personalized recommendations as we can see through Gini index and Shannon entropy values. In Last.fm, Categorical and Ontological settings show that our method outperforms the others whereas in Factual setting the results are almost identical. In terms of catalog coverage and distributional inequality our approach achieves good results. In Facebook Movies we see very a good improvement in terms of accuracy as it almost doubles up the ABItem-kNN algorithm values. Diversity results show no relevant differences between kaHFM and ABItem-kNN. In Facebook Music the accuracy improvements are clear in Categorical and Factual settings while the Ontological setting seems to be the least descriptive setting because accuracy results come reduced w.r.t. the other settings, and they are quite similar to ABItem-kNN. Finally, in Facebook Books, kaHFM shows the best results for all the considered metrics. Let us discuss the baselines more related to our approach. We compared kaHFM against ABItem-kNN to check if the collaborative trained features may lead to better similarity values. This hypothesis seems to be confirmed since in former experiments kaHFM beats ABItem-kNN 16 times over 18. This suggests that collaborative trained features achieve better accuracy results. Moreover, we want to check if a knowledge-graph-based initialization of latent factors may improve the performance of Factorization Machines. kaHFM beats BPR-FM 18 times over 18, and in our opinion, this happens since the random initialization takes a while to drive the Factorization machine to reach good performance. Finally, we want to check if collaborative trained features lead to better accuracy results than a purely informativeness-based Vector Space Model even though it is in its knowledge-graph-aware version. This seems to be confirmed in our experiments, since `kaHFM` beats *VSM* 15 times over 18.

In order to strengthen the results we got, we computed recommendations with 0,1,5,10,15,30 iterations. For the sake of brevity we report here only the plots related to Categorical setting shown in Figure 7.3. Results of the full experiments are available online[14].

It is worth to notice that in every case we considered, we show the best performance in one of this iterations. Moreover, the positive influence of the initialization of the feature vectors is particularly evident in all the datasets, with performances being very similar to the ones depicted in [218]. Given the obtained results we may say that the answer to RQ1 is positive when adopting `kaHFM`.

7.4.4 Semantic Accuracy

The previous experiments showed the effectiveness of our approach in terms of accuracy, diversity and novelty. In practical terms, we proved that: (i) content initialization generally lead to better performance with our method, (ii) the obtained items vectors are fine-tuned better than the original ones for a *top-N* item recommendation task, (iii) results may depend on the features we extract from the Knowledge Graph. However, we still do not know if the original semantics of the features is preserved in the new space computed after the training of `kaHFM` (as we want to assess by posing RQ2). In Section 7.3.5 we introduced `Semantics Accuracy (`*SA@nM*`)` as a metric to automatically check if the importance computed by `kaHFM` and associated to each feature reflects the actual meaning of that feature.

Thus, we measured `SA@`*nM* with $n \in \{1,2,3,4,5\}$ and $M = 10$, and evaluated the number of ground features available in the top-*nM* elements of $\mathbf{v}_i$ for each of the six datasets.

Table 7.8 shows the results for all the different datasets computed in the Categorical setting. In general, the results we obtain are noteworthy. We now examine the worst one to better understand the actual meaning of the values we get. In `Ya-`

[14] `https://github.com/sisinflab/papers-results/tree/master/kahfm-results/`

Semantics Accuracy	@M	@2M	@3M	@4M	@5M	F.A.
`Yahoo!Movies`	0.847	0.863	0.865	0.868	0.873	12.143
`LibraryThing`	0.960	0.996	0.998	0.999	0.999	3.820
`Last.fm`	0.960	0.987	0.991	0.994	0.995	6.615
`Facebook Music`	0.892	0.948	0.962	0.970	0.974	7.113
`Facebook Movies`	0.864	0.883	0.889	0.894	0.899	12.856
`Facebook Books`	0.995	1	1	1	1	3.133

Table 7.8: Semantics Accuracy results for different values of M. F.A. denotes the Feature Average number per item.

`hoo!Movies` Categorical setting, 747 different features compose each item vector (see Table 7.5). After the training phase, on average, more than 10 (equal to 0.847×12.143) over 12 features (last column in Table 7.8) are equal to the original features list. This means that `kaHFM` was able to compute almost the same features starting from hundreds of them. Even then, given the obtained results we may provide a positive answer to RQ2.

7.4.5 Generative Robustness

The previous experiment showed that the features computed by `kaHFM` keep their original semantics if already present in the item description. In section 7.3.6, we introduced a procedure to measure the capability of `kaHFM` to compute meaningful features. Here, we computed `1-Rob@nM` for the six adopted datasets. Results are represented in Table 7.9.

1-Robustness	@M	@2M	@3M	@4M	@5M	F.A.
`Yahoo!Movies`	0.487	0.645	0.713	0.756	0.793	12.143
`LibraryThing`	0.275	0.481	0.554	0.597	0.632	3.820
`Last.fm`	0.125	0.281	0.346	0.394	0.430	6.615
`Facebook Music`	0.714	0.893	0.935	0.955	0.966	7.113
`Facebook Movies`	0.821	0.945	0.970	0.980	0.984	12.856
`Facebook Books`	0.315	0.516	0.605	0.682	0.745	3.133

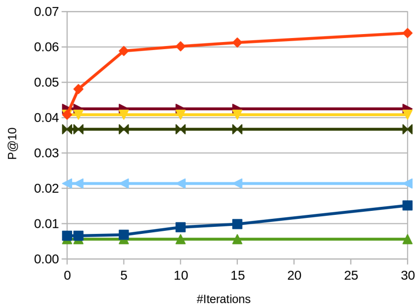
Table 7.9: 1-Robustness for different values of M. Column F.A. denotes the Feature Average number per item.

Even here, we focus on the CS setting. For a better understanding of the obtained results, we start by focusing on Yahoo!Movies which apparently has a bad behavior. As we said before, Table 7.8 shows that kaHFM was able to guess 10 on 12 different features for Yahoo!Movies. In this experiment, we remove a feature thus making kaHFM to guess an average of 9 over 12 features. What we measure now is if kaHFM is able to guess the removed feature in the remaining 3 "slots". Results in Table 7.9 show that our method is able to put the removed feature in one of the three slots the 48.7% of the times over 747 overall features. This example should help the reader to appreciate even more Facebook Music and Facebook Movies results. For the remaining datasets the situation is even much harder because there were no free slots (see Table 7.8). Thus, our method has only one missing slot to fill with the right feature. Let us take Facebook Books as an example: there are 263 different features in the item vector (see Table 7.5) and a very low average number of features per item (3.133). kaHFM is able to fill the missing slot with the right feature 31% of the times. The obtained results shows that kaHFM is able to propose meaningful features as we asked with RQ3.

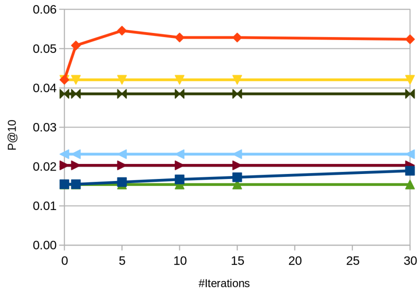
7.5 Conclusion

In this work, we have proposed an explainable method, kaHFM, in which we bind the meaning of latent factors for a Factorization machine to data coming from a knowledge graph. We evaluated kaHFM on six different publicly available datasets and compared it against state-of-the-art algorithms showing that our approach outperforms the other approaches with respect to accuracy, diversity, and novelty on different sets of semantics-aware features. In particular, we considered Ontological, Categorical and Factual information coming from a freely available knowledge graph. We have shown that the generated recommendation lists are more precise and personalized, and they select more items from the long tail. Summing up, performed experiments show that: (RQ1) the learned model shows very good performance in terms of accuracy, novelty and diversity and, at the same time, is effectively explainable; (RQ2) the computed features are semantically meaningful; (RQ3) the model is robust regarding computed features.

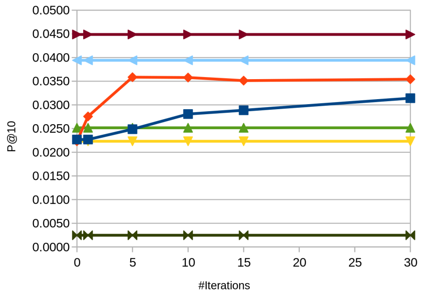
In the future we want to test the `kaHFM` performance in different scenarios, other than recommender systems. Moreover, the model can be improved in many different ways. First of all, a stopping condition based on a validation set could be introduced to avoid wasteful training steps. Different relevance metrics could be beneficial in different scenarios, as the method itself is agnostic to the specific adopted measure. This work focused on the items' vector; however, an interesting key point would be analyzing the learned users' vectors to extract more accurate profiles. Furthermore, it would be useful to exploit `kaHFM` in order to provide suggestions to knowledge graphs maintainers while adding relevant missing features to the knowledge base. In this direction, we would like to evaluate our approach in knowledge graph completion tasks.



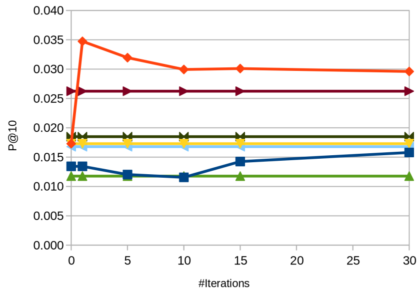
(a) Precision on `LibraryThing`



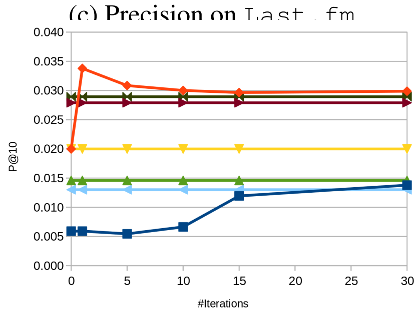
(b) Precision on `Yahoo!Movies`



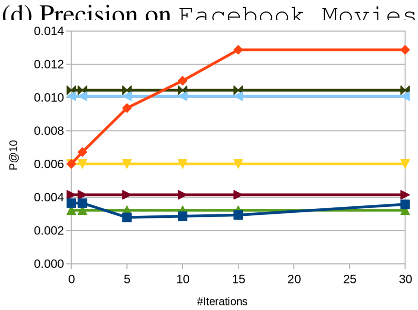
(c) Precision on `Last.fm`



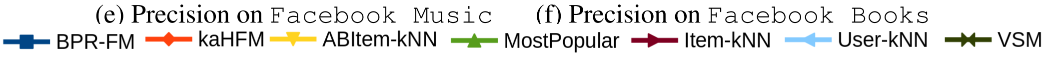
(d) Precision on `Facebook Movies`



(e) Precision on `Facebook Music`



(f) Precision on `Facebook Books`



(g) legend

Figure 7.3: Precision@10 varying # iterations 0, 1, 5 , 10 , 15, 30

Chapter 8

Reasoning with Preferences

8.1 Introduction

In this line of research, the focus is on model-based preference reasoning, which relies on specific assumptions about the structure of the preference relation [102]. The simplest assumption that can be made is that the target ranking of a set of resources, described in terms of multiple attributes, can be represented as a lexicographical order [79]. Lexicographical preference models define orders of importance on the attributes that describe the objects of a domain. As an example, consider the choice of a movie. Typically, the most important attribute that one considers is the genre of the movie (e.g., drama, superhero, etc.). Then, among the movies of the preferred genre, the choice can rely on the movie's actor (e.g., Tom Hanks, Christian Bale, etc.). The assumption of a lexicographical order restricts significantly the hypothesis space, but induces a bias rarely justified in practical applications. In fact, preferences on individual attributes are generally not independent of each other. With reference to the movie domain, Tom Hanks may be preferred to Christian Bale, if the movie genre is drama, while Christian Bale may be preferred in case of a superhero film. *CP-nets* [44] offer a language to express preferences on the values of single attributes, and, at the same time, allow to model dependencies of this type. A CP-net is a qualitative graphical representation that reflects conditional dependence and independence of preferences under a *ceteris paribus* (all else being equal) interpretation. It is a compact representation of a complex preference relation (partial order), where each node refers to a single attribute and is associated with a function that assigns a preference relation on the values of that attribute to each combination of the values of the parent attributes. More precisely, CP-nets require that the user specifies (i) for any attribute A of interest, which other attributes can impact her preferences for values of A (the parents of A), and (ii) for each instantiation of the parent attributes, the preference ordering over values of A. Points (i) and (ii) could make CP-nets a rather rigid formalism, compared to the expressive needs of a user. Furthermore, some statements that are very natural for the user to assess cannot be represented within a CP-net. *Conditional preference theories* (or *CP-theories*) [280] are a more general and flexible formalism for qualitative preferences that allows to go beyond the expressiveness limitations of CP-nets.

In our previous work [226], we focused on the well-known CP-net graphical language and have addressed the problem of preference representation and reasoning with Linked Data from different perspectives. We have proposed a vocabulary to represent statements formulated according to the ceteris paribus semantics and have shown how to encode a CP-net by means of this vocabulary. Inspired by [111], we have also explained how to embed such a compact preference model into a SPARQL 1.1 query in order to access semantic data in a personalized way. The current investigation extends the leading motivation and the approach of a previous work [226], but embraces the more general and flexible formalism of CP-theories. We point out that the approach proposed here only deals with context-uniform conditional (cuc) acyclic CP-theories [280], which are a special type of CP-theories exposing nice polynomial computational properties while comparing outcomes.

This study heavily extends the approach presented in [226] in many points. First of all, here we deal with the much more expressive CP-theories instead of CP-nets. A new extended vocabulary as well as a completely new algorithm to encode

CP-theories in SPARQL is also proposed. Moreover, an implementation of the overall framework is presented together with experimental evaluations targeted at assessing the users' experience in representing their preferences as CP-theories and the performance of the tool. The main contributions of this work can be summarized as follows:

- presentation of `RDF` vocabularies to represent qualitative preference statements over Linked Data, built on top of the vocabulary proposed in [226], but adjusted for more general preference statements;
- an encoding into `RDF` triples of the qualitative preferential information represented by a CP-theory and the exploitation of theoretical results of [279] to compute a partial order over items for cuc-acyclic cases;
- a procedure to translate conditional preference statements into a `SPARQL` 1.1 query able to retrieve a **ranked list** of resources whose order reflects the user preferences;
- an application framework that meets a user's needs while representing her preferences as a CP-theory encoded in `RDF` and that eventually allows a `SPARQL`-enabled software agent to retrieve a ranked list of resources according to the users's tastes;
- An experimental evaluation to verify the effectiveness of the proposed approach.

The rest of the chapter is structured as follows. Section 8.3 presents the motivating scenario that fostered the overall approach. The semantics of CP-theories and a recap on some relevant results and theorems has been provided in background, Section 2.2.1. In Section 8.4, we propose an `RDF` vocabulary to represent a CP-theory with the preferential statements of a user, and then we show how to embed the `RDF` version of the CP-theory into a `SPARQL` query able to retrieve a ranked list of results ordered according to user's preferences. Section 8.7 describes the tool that supports the user both in the formulation of her preferences under the CP-theories semantics and in retrieving the resources of interest ordered according to her preferences. The results on a user study are reported in Section 8.8, where we also measure the performance of the implemented system on a synthetic dataset. Section 8.2 provides an overview of related work about preference reasoning and enabling query languages with preferences. Conclusions close the chapter.

8.2 Related work

The ability to infer, model, and reason with user preferences has been recognized as a prominent research direction in many fields, especially artificial intelligence (AI) [85, 212]. Preferences are generally classified as quantitative, if they make use of a scoring function to assess an order over the available resources, resulting in a total order, or qualitative, if they are treated independently, resulting in incomparable resources and a partial preference order. Much work has focused on qualitative approaches, since these are closer to how people express their preferences; among the earliest logic-based approaches is von Wright's [272]. Following the overview over qualitative multi-attribute preference reasoning approaches provided by [195], in AI, there are, in particular, (i) methods adopting graphical structures to represent and reason about preferences, e.g., CP-nets [44] and TCP-nets [45]; (ii) methods that extend constraints satisfaction problems and incorporate soft constraints, as in the approximation of CP-nets with soft constraints described in [83, 82]; and (iii) methods that use specific logic-based languages to represent qualitative preferences and derive utility functions, exploiting, e.g., machine learning techniques, such as support vector machines [81, 80].

Conceptually close in spirit to our investigation is in particular [188], where ontological knowledge expressed via existential rules in Datalog$^{\pm}$ is combined with CP-theories to represent qualitative conditional preferences along with domain knowledge, and to perform preference-based answering of conjunctive queries. Another related work [66] combines Datalog with CP-theories, but only considers atomic queries. Our work, in contrast, focuses on SPARQL queries in a more restricted ontological context and conditional preferences specified via cuc-acyclic CP-theories.

There is also a large body of work on handling preferences in logic programming, e.g., asprin [48], which is a framework for handling preferences among the stable models of a logic program. Similarly, the qualitative choice logic [47] is a propositional logic for representing a preference relation among models, which allows to specify alternative, ranked options for problem solutions. The above two and similar works on handling preferences in logic programming are fundamentally different from our approach, as they are about preferences for ordering models of a logic program, rather than preferences for ordering the answers to a query subject to all models of a knowledge base.

Databases are another research area where preferences have been investigated. In a relational database management system, for example, the *top-k* (or *ranking*) queries represent a quantitative approach, since they return the *k* best matches according to a numerical score. In [158], a formalism supporting ranking queries for a relational database is presented. With reference to the qualitative approach instead, *skyline* queries [41] extend the notion of best matching to contexts, where multiple independent scores have to be taken into account. The result of a *skyline* query is a set of objects that are no worse than any other across all dimensions of a set of independent Boolean or numerical preferences [41]. Within the database community, both Chomicki [63, 64] and independently Kießling and colleagues [143, 144] formalized the first examples of *preference-based querying* languages, that is, extensions of SQL that support the specification of quantitative and qualitative queries.

The notion of preference is of primary importance also in the Linked Open Data context. The provision of means to enable users to look for data sources (e.g., `SPARQL` endpoints) and data content that is tailored to their individual preferences is one of the target of the original project by Tim Berners-Lee et al. Even the motivating example proposed in the introductory article about the Semantic Web [38] can be interpreted as a preference-based search, as extensively discussed in [248]. Based on this insight, in [248], the authors add preference-based querying capabilities to the most known Semantic Web query language, `SPARQL`. However, when the paper was published, it was not possible to specify multiple (independent) preference dimensions in `SPARQL`, and consequently the authors had to introduce the PREFERRING solution modifier. For example, the following query provides a preference-enabled SPARQL query for a user who is searching for an appointment, preferring excellent therapist, appointments out of the rush hour and later appointments over earlier ones, if both are equal with respect to the rush hour constraint.

```
1 SELECT ?appointment WHERE {
2 ?terapist :rated ?rating;
3 :offers ?appointment.
4 ?appointment :starts ?start;
5 :ends ?end.
6 PREFERRING (?rating = excellent AND
7 ?end < 1600 || ?start > 1800
8 CASCADE HIGHEST(?start))
9 }
```

At line 6, the PREFERRING clause behaves as a solution modifier, and the AND keyword separates independent preference dimensions. The CASCADE keyword at line 8 allows to give higher priority to the left-hand preference over the right-hand one. In their paper, the authors state that within the same SPARQL query, the use of a LIMIT k statement in combination with PREFERRING ones could inform the query evaluator to go deeper in the retrieval of skyline solutions, thus allowing the system to return a set of results ordered by user preferences.

A mapping operation between an OWL ontology, called *OWLPref*, and the SPARQL Preference syntax of [248] has been proposed by [155]. OWLPref allows for representing in a declarative, domain-independent and machine-interpretable way several kinds of preferences, namely, *SimplePreference*, *CompositePreference* (which makes compositions of the former), and *ConditionalPreference* (which models preferences that vary according to the context, thanks to a property *OnCondition*). However, considering the unavailability of conditional preferences in the SPARQL Preference syntax of [248], the use of *ConditionalPreference* in OWLPref seems of marginal utility.

The *PrefSPARQL* syntax of [111] keeps the goal of identifying the Pareto-optimal set, but introduces preferences at the level of filters. It still uses the AND to separate independent dimensions and to build what the authors call *MultidimesionalPref*. Each "dimension" is either a *conditional* preference (IF-THEN-ELSE) or an *atomic* preference, which in turn can be a simple expression or can involve more complex constructs [111]. Besides the support for conditional preferences and the substitution of CASCADE with PRIOR TO, the main innovative point of [111] with respect to [248] is perhaps that the proposed preference-enabled query can be completely rewritten using SPARQL 1.1 characteristics. In particular, [111] uses the FILTER NOT EXISTS. The translation of the previous query according to the *PrefSPARQL* query rewriting is given below.

```
1 SELECT ?appointmentA WHERE {
2 ?terapistA :rated ?ratingA;
3 :offers ?appointmentA.
4 ?appointmentA :starts ?startA;
5 :ends ?endA.
6 BIND ((?ratingA = :excellent) AS ?Pref1A)
7 BIND ((?endA < 16 || ?startA > 18:00) AS ?Pref2A)
8 BIND ((?startA) AS ?Pref3A)
9 FILTER NOT EXISTS {
10 ?terapistB :rated ?ratingB;
11 :offers ?appointmentB.
12 ?appointmentB :starts ?startB;
13 :ends ?endB.
14 BIND ((?ratingB = :excellent) AS ?Pref1B)
15 BIND ((?endB < 1600 || ?startB > 1800) AS ?Pref2B)
16 BIND ((?startB) AS ?Pref3B)
17 FILTER (
18 ((?Pref1B > ?Pref1A)  &&
19 !((?Pref2B < ?Pref2A) ||
20 (?Pref3B < ?Pref3A && ?Pref2B = ?Pref2A)))
21 ||
22 (!(?Pref1B < ?Pref1A)  &&
23 ((?Pref2B > ?Pref2A) ||
24 (?Pref3B > ?Pref3A && ?Pref2B = ?Pref2A))))}
25 }
```

The query looks for appointments ?appointmentA satisfying a certain pattern expressed in lines 2–5. The research is carried out checking that there is no ?appointmentB that verifies the same pattern (lines 10–13) and dominates ?appointmentA in any preference dimension. The example refers to only two independent preference dimensions and the situations when ?appointmentB dominates ?appointmentA are represented in the branches of || symbol at line 21, that is, lines 18–20 and lines 22–24. For the sake of completeness, ?appoint-

`mentB` would dominate `?appointmentA` if it was better in one dimension (line 18 or line 23–24) and no worse in the other one (line 19–20 or line 22). The `PRIOR TO` preference relation is encoded in lines 19–20 and 23–24 through the `||` operator.

Although *PrefSPARQL* allows the user to encode conditional preferences in a SPARQL 1.1 query, it differs from the approach that we presented here in at least three main aspects: (i) in *PrefSPARQL*, the focus is on computing the most preferred solution (undominated outcome), given a set of conditional preferences, while we provide a list of results ordered by user preferences; (ii) they deal with conditional preferences in the form $u_\varphi : x_\varphi > x^{'}_\varphi$, while our approach is able to manage CP-statements in the form $u_\varphi : x_\varphi > x^{'}_\varphi\,[W_\varphi]$, which result to be much more expressive for finite and discrete domains even in their cuc-acyclic version that we consider here; (iii) we provide an ontological vocabulary and a procedure to automatically encode preferences in a SPARQL query. However, thanks to its more agile structure, differently from our approach, *PrefSPARQL* allows the user to express preferences on variables with continuous domains as well as the usage of comparison operators.

In [265], the authors present SPREFQL, an extension of the SPARQL language that allows for appending a "PREFER" clause, which expresses 'soft' preferences over the query results obtained by the main body of the query. The main ideas behind the approach are to associate relations of tuples with preference formulas, and to select the relations' most preferred tuples via a so-called winnow operator. Consequently, the approach does not allow for expressing conditional ceteris paribus preferences as in CP-theories.

As a general remark, previous SPARQL-related works on preference reasoning have been mainly devoted to preference representation and the retrieval of undominated outcomes. In principle, one may encode each of them in a procedural approach able to compute the first level of undominated solutions, then the second one and so one. At each iteration step, the procedure should be able to filter out the results coming from the "higher levels" of results computed in the previous steps.

Less closely related are approaches to preference-based query answering over graph databases. In particular, [100] presents regular languages for graph queries, where answers are partially ordered via a partial order on the strings of the languages. In the same vein, [109] introduces preferential regular path queries for enhanced querying of semi-structured databases. Query symbols are annotated with preference weights for scaling up or down the importance of matching a symbol against a database edge label. The paper studies (progressive) query answering, (certain) query answering in LAV data-integration systems, and query containment and equivalence. A similar approach in [97] introduces a graph query language that enables to declaratively express preferences. None of the above approaches to preference-based query answering over graph databases (which are intuitively based on (potentially recursive) pattern-recognition-style regular expressions), however, allows for expressing conditional ceteris paribus preferences as in CP-theories. Less closely related are also information retrieval systems based on manipulating fuzzy truth values (which may also be interpreted as quantitative preferences), such as the fuzzy multimedia retrieval system in [90].

8.3 Motivating scenario

The leading scenario behind the framework that we propose here is that of a distributed system where a user may pose a query to a `SPARQL` endpoint and have the returned results ordered with respect to a set of personal preferences on a specific knowledge domain. For instance, a user might be willing to get a list of books to read by querying `DBpedia` and then have it ranked according to a set of preferences hosted on their own Web page. A possible implementation of the approach that we propose is depicted in Fig. 8.1,[1] where the main building blocks required to implement the whole framework are shown:

- a reference model to encode and reason with preferences (CP-theory in our case);
- an ontological vocabulary to represent preferences by adopting Web languages;

[1]For ease of presentation, we use `DBpedia` as the main dataset to query. The approach can be adapted to any Linked Data dataset.

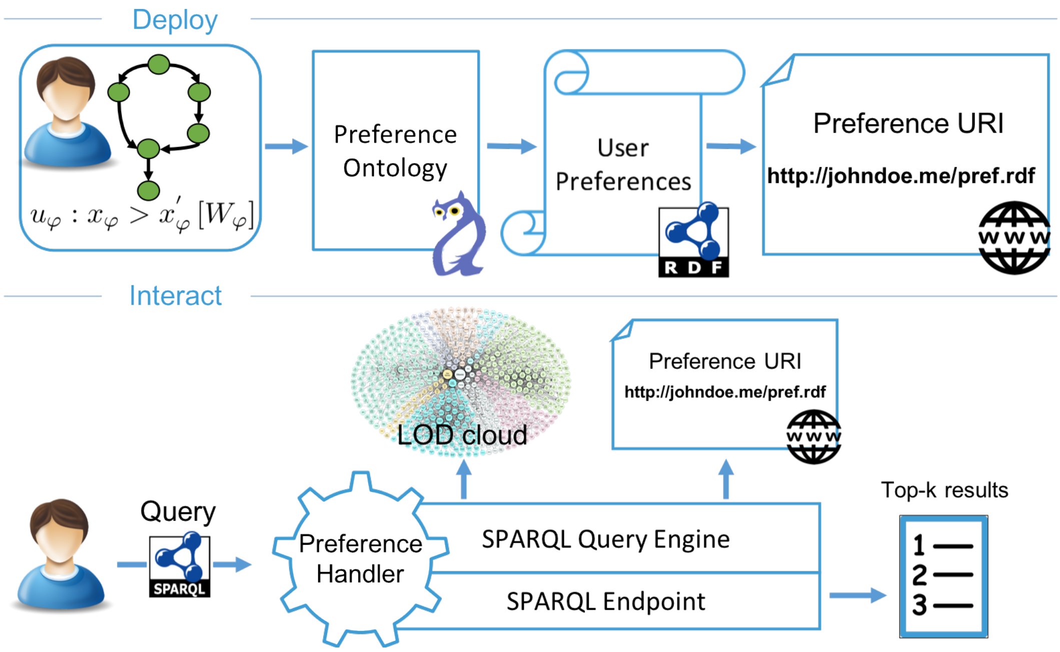


Figure 8.1: A graphical representation of the proposed approach. The Deploy and Interact steps only rely on the adoption of standard technologies and languages.

- a tool able to handle and manage preferences as well as to encode them in a set of `SPARQL` queries.

In order to implement the whole framework, we propose the following deployment and interaction steps.

Deploy.

- *Model user preferences*. We adopt CP-theories [279] as reference model to represent user preferences. As we will see in Section 2.2.1, they are a formalism to represent and reason with sets of qualitative preferences.[2]
- *Encode preferences in `RDF` by means of a preference ontology*. We developed an OWL ontology to represent CP-theory statements encoding user preferences (see Section 8.4).

[2]The interest here is not whether such preferences are automatically learned from data or manually modeled and set by a human agent.

- *Publish user preferences publicly on the Web*. The aim is to foster a principled adoption of user preferences by systems interested in providing a personalized access to data available in the Linked Data cloud. User preferences can be encoded and published in an RDF file.

Interact.

- *Use `SPARQL` to get user preferences*. A preference handler loads[3] the preference model of the user encoded in `RDF` by means of a general purpose `SPARQL` query engine.
- *The preference handler formulates `SPARQL` 1.1 queries able to retrieve and order resources by taking into account user preferences*. Standard `SPARQL` queries are formulated in order to rank the result set with reference to the preferences expressed by the user.

A strong requirement that we had in mind while developing our solution was to use only standard Web technologies and languages to implement the overall framework. Indeed, we selected `RDF` to model user preferences and the power of `SPARQL` 1.1 to perform preference-based reasoning in order to rank the results of a query. In fact, as we will see in Section 8.4.2, advanced features of the last version of the `SPARQL` query language can be employed to perform preference reasoning over a model based on CP-theories. In Section 8.3, all the details needed to implement the Deploy and Interact steps in a pure Linked Data setting will be provided.

8.4 CP-theories and Linked Open Data

As we stated in Section 8.1, the target of this work is twofold. On the one hand, we want to supply the user with a vocabulary to represent qualitative statements formulated in terms of ceteris paribus semantics. On the other hand, we aim to provide an encoding of user preferences that can be used in a top-k query answering scenario. In this section, we start by proposing a first ontology that allows a system to

[3]See *SPARQL 1.1 Update* specification at `https://www.w3.org/TR/sparql11-update/\#load`.

represent preferential statements according to CP-theories in a very straightforward way and an extended version to manage the directed graph $J_o(\Gamma)$ on V introduced in Section 2.2.1 and exploited in Theorem 2. Note that RDF triples encoding the directed graph can be automatically derived from the original preferential statements. In Section 8.7, we provide a description of an implemented tool to infer the RDF version of the directed graph, starting from a set of conditional preferences. We deal with the complementary target in Section 8.4.2, where we show how to employ a user profile represented as an instantiation of the extended ontology to encode the corresponding preferences in a standard SPARQL query able to retrieve and rank resources in a personalized way.

Figure 8.2 shows the ontology that we modeled to express user profiles in terms of CP-theory statements[4]. The main idea behind the modeling of the ontology is that we may express preferences on properties of items that the user is looking for, e.g., `dbo:literaryGenre`, `dbo:country`, `dbo: subsequentWork`, or potentially `dbo:filmVer-sion`. Hereafter, the ontology in Figure 8.2 will be referred to as the *lite* ontology. The aim of the *lite* ontology is that of creating an ontological vocabulary providing all the elements to syntactically represent conditional preference statements in a theory Γ. By means of this ontology, it is possible to encode whatever preference φ in its general form $u_\varphi : x_\varphi > x_\varphi^{'} [W_\varphi]$.

The ontology is composed by four main classes and nine properties. The class `Value` represents possible values of a variable. If we look at the book *GoodKnyght!* in Example 5, we see that the "actual values" for which the user expresses a preference are composed by both a *property* (e.g., `dbo:country`) and its related *object* (e.g., `db:United_Kingdom`). This is the reason why the class `Value` is the domain of the two properties `attribute` and `value`. The former mapping the property, the latter mapping the object. `Condition` is used to express the conditional part u_φ of a preference statement $u_\varphi : x_\varphi > x_\varphi^{'} [W_\varphi]$, which is also the condition for the relative importance [45] of the variable X_φ over variables in W_φ in case $W_\varphi \neq \emptyset$. It is the domain of the property `contains`, whose range

[4]The corresponding OWL file is available at `http://sisinflab.poliba.it/semanticweb/lod/ontologies/cpt_light.owl`

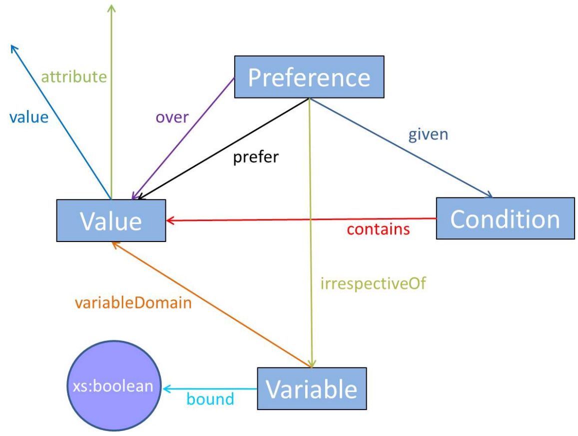


Figure 8.2: A graphical representation of the *lite* version of the ontology proposed to represent conditional statements.

is `Value`. The class `Preference` represents the whole conditional statement φ. The properties having `Preference` as domain reflect the structure of the preferential statement "`given` a `Condition`, I `prefer` a `Value` `over` another `Value`, optionally `irrespectiveOf` some other variables". The class `Variable` is used to model the variables of a CP-theory, and it is domain of `variableDomain`, whose range is `Value`. Finally, we have the `bound` property needed to explicitly state if the `value` associated to an `attribute` is an actual value (as for `dbo:country` and `dbo:literaryGenre`) or if it represents the situation that we (do not) have a triple involving the `attribute`, as for `dbo:subsequentWork` or `dbo:filmVersion`. Here, we adopted the modeling choice of representing directly the conditions generated by the combination for the values of variables in U_φ instead of relating the variables themselves. As an example, in case we have "*Given an English book that is part of a saga, I prefer...*", with reference to the previous example, the corresponding encoding will be[5]

```
cpl:combined-cond a cpl:Condition ;
cpl:contains  cpl:c1 ;
cpl:contains  cpl:s1 .
```

[5]Here, we use the prefix `cpl` to denote `<http://sisinflab.poliba.it/semanticweb/lod/ontologies/cpt_light.owl\#>`

Here, `cpl:combined-cond` represents u_φ as a whole, while `cpl:c1` and `cpl:s1` represent the values x_1 and x_2 composing $u_\varphi = x_1 x_2$.
We will see how this modeling choice will be useful when embedding the CP-theory into a `SPARQL` query.

Notice that such classes and predicates are sufficient for the user to express the CP-theory with her preferential statements, and to facilitate the user experience even further, we built a user-friendly tool, where preferences related to a specific domain (e.g., books or movies) can be added. The aforementioned tool will be extensively described in Section 8.7.

Example 5 (Books cont'd)
With respect to Giorgio's preference "*given an English book, he prefers those being part of a saga*", if we look in `DBpedia`, we may find, for instance, the book *GoodKnyght!*. Indeed, we have:

```
@prefix db: <http://dbpedia.org/resource/>
@prefix dbo: <http://dbpedia.org/ontology/>

db:GoodKnyght! a dbo:Book ;
dbo:country db:United_Kingdom ;
dbo:subsequentWork db:Whizzard! .
```

From the previous `RDF` statements, we see that Giorgio's preference refers to values of objects in a triple with reference to a specific predicate. Indeed, given a set of resources of type `dbo:Book` such that the value for `dbo:country` is `db:United_Kingdom`, he prefers those with an associated triple whose predicate is `dbo:subsequentWork`. In order to be fully compliant with the Linked Data technological stack, we need a vocabulary/ontology that allows users to represent their preferences on different attributes of resources that they might be interested in. Hence, with reference to the ontology in Fig. 8.2, we have the following `RDF` triples modeling the preference introduced at the beginning of this example.

```
@prefix cpl: <http://sisinflab.poliba.it/
semanticweb/lod/ontologies/cpt_light.owl#>
@prefix db: <http://dbpedia.org/resource/>
```

```
@prefix dbo: <http://dbpedia.org/ontology/>

### Variables ###
cpl:country a cpl:Variable;
cpl:bound true;
cpl:variableDomain cpl:c1,cpl:c2.

cpl:subsequentWork a cpl:Variable;
cpl:bound false;
cpl:variableDomain cpl:s1, cpl:s2.

### Values allowed for each variable ###
cpl:c1 a cpl:Value;
cpl:attribute dbo:country;
cpl:value db:United_Kingdom.

cpl:c2 a cpl:Value;
cpl:attribute dbo:country;
cpl:value db:France.

cpl:s1 a cpl:Value;
cpl:attribute dbo:subsequentWork;
cpl:value cpl:subsequentWorkYes.

cpl:s2 a cpl:Value;
cpl:attribute dbo:subsequentWork;
cpl:value cpl:subsequentWorkNo.

### Condition ###
cpl:cond a cpl:Condition;
cpl:contains  cpl:c1.

### Preference ###
cpl:pref a cpl:Preference;
cpl:given cpl:cond;
cpl:prefer cpl:s1;
cpl:over cpl:s2.
```

■

Once we have defined and modeled Γ in RDF, in order to compare two outcomes α and β as in Theorem 2, we may build the RDF version of the directed graph $\triangleright_\alpha$ on V (see Section 2.2.1).

To this aim, the *lite* ontology is extended as shown in Figure 8.3[6] to what we call the *full* ontology. Hence, starting from a set of preferences represented via the *lite* ontology, we derive its *full* version such that, once instantiated with an outcome α, it represents $\triangleright_\alpha$. The derivation step is performed by adding new statements via the following relations:

`moreImportantThan` has `Variable` both as domain and as range, and it is used to model the transitive closure of dependency and (unconditional) relative importance information, i.e., for the transitive closure of edges $U_\varphi \to \{X_\varphi\} \cup W_\varphi$, and, if $U_\varphi = \emptyset$, for $\{X_\varphi\} \to W_\varphi$, for any φ.

`conditionallyMoreImportantThan` takes into account conditional relative importance information. Its range is an instance of the new class `InstanceOfRelativeImportance`, which is used to represent a pair (u_φ, Z), $Z \in W_\varphi$, for a statement φ with $U_\varphi \neq \emptyset$.

In fact, the class `InstanceOfRelativeImportance` is the domain of the property `hasCondition`, whose range is the `Condition` instance representing u_φ, and of the `hasLessImportantVariable` property, whose range is the `Variable` instance for Z. In the following, we say "X is `conditionallyMoreImportantThan` Y under the `Condition` C" when X is linked via `conditionallyMoreImportantThan` to an instance of the class `InstanceOfRelative-Importance`, which, in turn, is linked by `has-Condition` to a `Condition` C and by `has - LessImportant-Variable` to a `Variable` Y.

Both properties must act as transitive relations. Therefore, we add more statements to the original preferences by means of the following rules involving the

[6]The corresponding OWL file is available at `http://sisinflab.poliba.it/semanticweb/lod/ontologies/cpt_full.owl`

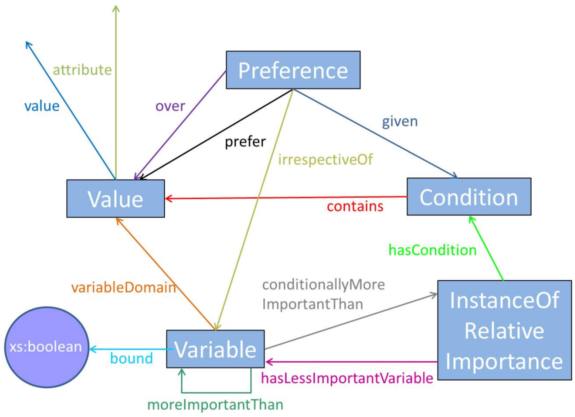


Figure 8.3: A graphical representation of the *full* ontology proposed to represent conditional statements.

transitive closure of property `moreImportantThan` and then, for pairs of variables (Y,Z) not linked by it, the transitive closure of property `conditionally-MoreImportantThan`.

- if variable Y is `moreImportantThan` a variable X and at the same time variable X is `conditionallyMoreImportantThan` a variable Z, under a condition C not involving values of Y, then we add the RDF statements representing that "Y is `conditionallyMoreImpor-tantThan` Z under the condition C";

- if Y is `conditionallyMoreImportantThan` a variable X under a condition C, X is at the same time `moreImportantThan` a variable Z, and the condition C does not involve any value of Z, then the additional fact we add is "Y is `conditionallyMoreImportantThan` Z under the condition C";

- if Y is `conditionallyMoreImportantThan` a variable X under a condition C, and X is `conditionallyMoreImportantThan` a variable Z under a condition C', then the additional fact added to the knowledge base is "Y is `conditionallyMoreImportantThan` Z under the condition C''", where $C'' = C \wedge C'$ is the condition joining C and C', but only if $C \wedge C'$ does not contains values of Y, Z or two different values of any other variable.

According to the properties just introduced, given a pair of outcomes (α, β), if there exist no variable $X^{'} \in \Delta(\alpha,\beta)$ that is `moreImportantThan` X and no variable $X^{''} \in \Delta(\alpha,\beta)$ that is `conditionallyMore-ImportantThan` (u, X), with α extending u, then a variable X is in $\Theta^{'}(\alpha,\beta)$.

Example 6 (Books cont'd)
The *full* encoding corresponding to $\Gamma_{\mathrm{C-LG-SW-F}}$ for Giorgio's preferences in Table 2.1, is represented in Listing 8.1[7]. ■

```
@prefix cpt: <http://sisinflab.poliba.it/semanticweb/
lod/ontologies/cpt_full.owl#>
@prefix db: <http://dbpedia.org/resource/>
@prefix dbo: <http://dbpedia.org/ontology/>

cpt:country1 a cpt:Value;
cpt:attribute dbo:country;
cpt:value db:United_Kingdom.
cpt:country2 a cpt:Value;
cpt:attribute dbo:country;
cpt:value db:France.
cpt:genre1 a cpt:Value;
cpt:attribute dbo:literaryGenre ;
cpt:value db:Crime_fiction.
cpt:genre2 a cpt:Value;
cpt:attribute dbo:literaryGenre ;
cpt:value db:Autobiographical_novel.
cpt:sw1 a cpt:Value;
cpt:attribute dbo:subsequentWork;
cpt:value cpt:subsequentWorkYes.
cpt:sw2 a cpt:Value;
cpt:attribute dbo:subsequentWork;
cpt:value cpt:subsequentWorkNo.
cpt:film1 a cpt:Value;
cpt:attribute dbo:filmVersion;
cpt:value cpt:filmVersionYes.
cpt:film2 a cpt:Value;
cpt:attribute dbo:filmVersion;
cpt:value cpt:filmVersionNo.

cpt:conditionC1 a cpt:Condition;
cpt:contains  cpt:country1.
cpt:conditionC2 a cpt:Condition;
cpt:contains  cpt:country2.
cpt:conditionSW1 a cpt:Condition;
cpt:contains  cpt:sw1.
cpt:conditionSW2 a cpt:Condition;
cpt:contains  cpt:sw2.

cpt:country a cpt:Variable;
cpt:bound true;
cpt:variableDomain cpt:country1,cpt:country2;
```

[7]For conciseness, the prefix `cpt` is always assumed in the following for `<http://sisinflab.poliba.it/semanticweb/lod/ontologies/cpt_full.owl\#>`.

```
cpt:moreImportantThan cpt:literaryGenre;
cpt:moreImportantThan cpt:subsequentWork;
cpt:moreImportantThan cpt:filmVersion.
cpt:literaryGenre a cpt:Variable;
cpt:bound true;
cpt:variableDomain cpt:genre1,cpt:genre2;
cpt:conditionallyMoreImportantThan
cpt:instanceOfRelativeImportance1;
cpt:conditionallyMoreImportantThan
cpt:instanceOfRelativeImportance3.
cpt:subsequentWork a cpt:Variable;
cpt:bound false;
cpt:variableDomain cpt:sw1, cpt:sw2;
cpt:moreImportantThan cpt:filmVersion;
cpt:conditionallyMoreImportantThan
cpt:instanceOfRelativeImportance2.
cpt:filmVersion a cpt:Variable;
cpt:bound false;
cpt:variableDomain cpt:film1, cpt:film2.

cpt:instanceOfRelativeImportance1
a cpt:instanceOfRelativeImportance;
cpt:hasCondition cpt:conditionC1;
cpt:hasLessImportantVariable cpt:subsequentWork.
cpt:instanceOfRelativeImportance2
a cpt:instanceOfRelativeImportance;
cpt:hasCondition cpt:conditionC2;
cpt:hasLessImportantVariable cpt:literaryGenre.
cpt:instanceOfRelativeImportance3
a cpt:instanceOfRelativeImportance;
cpt:hasCondition cpt:conditionC1;
cpt:hasLessImportantVariable cpt:filmVersion.

cpt:preference1 a cpt:Preference;
cpt:prefer cpt:country2;
cpt:over cpt:country1.
cpt:preference2 a cpt:Preference;
cpt:given cpt:conditionC1;
cpt:prefer cpt:genre2;
cpt:over cpt:genre1;
cpt:irrespectiveOf cpt:subsequentWork.
cpt:preference3 a cpt:Preference;
cpt:given cpt:conditionC2;
cpt:prefer cpt:sw2;
cpt:over cpt:sw1;
cpt:irrespectiveOf cpt:literaryGenre.
cpt:preference4 a cpt:Preference;
cpt:given cpt:conditionC1;
cpt:prefer cpt:sw1;
cpt:over cpt:sw2.
cpt:preference5 a cpt:Preference;
cpt:given cpt:conditionC2;
cpt:prefer cpt:genre1;
cpt:over cpt:genre2.
cpt:preference6 a cpt:Preference;
cpt:given cpt:conditionSW1;
cpt:prefer cpt:film1;
cpt:over cpt:film2.
cpt:preference7 a cpt:Preference;
cpt:given cpt:conditionSW2;
cpt:prefer cpt:film2;
cpt:over cpt:film1.
```

Listing 8.1: The `RDF` version of the CP-theory $\Gamma_{\mathrm{C-LG-SW-F}}$ in Table 2.1, according to the *full* ontology.

In Section 8.4.2, we will see how to pose a SPARQL query against the *full* version of Γ in order to compute $\gg_\Gamma$, thus retrieving a ranked list of semantic resources ordered according to a user's preferences.

8.4.1 The special case of CP-nets

Before moving to the description of how to encode a CP-theory in a `SPARQL` query for personalized results ranking, we point out that the ontological model just described subsumes the vocabulary introduced in [226] to represent CP-nets models. There, we proposed how to model the information encoded in a CP-net through an ontology and how to formulate the query able to order outcomes accordingly in a consistent way. On the other hand, we know that CP-nets are a special and simple case of CP-theories and, therefore, we want to show how the new ontology in Figure 8.3 can deal with a CP-net.

The theoretical result exploited in [226] (namely Corollary 4 of [44]) orders an outcome o over another one o', consistently with a CP-net $\mathcal{N}$, if there exists a variable X such that o and o' assign the same values to all ancestors of X in $\mathcal{N}$ and given the assignment provided by o (and o') to the parents of X, i.e., $Pa(X)$, o assigns a more preferred value to X than that assigned by o' (according to the conditional preference table of X). The sufficient condition of Corollary 4 of [44] may be reformulated asking for a variable X such that there does not exist any variable `moreImportantThan` X different in o and o', and, given the assignment provided by o (and o') to $Pa(X)$, o assigns a more preferred value to X than that assigned by o'. The predicate `moreImportantThan`, in fact, covers dependency information and, when applied to a CP-net, allows to define a set of variables coincident with the ancestor set. Moreover, for a CP-net, the predicate `given` for any instance of `Preference` ordering the values of a variable may be used to define the parent set of that variable. The ontological model proposed in [226] for CP-nets is hence subsumed by the *full* ontology of Figure 8.3. On the other hand, if one wants to represent a CP-net according to the *full* ontology, one has to consider that the dependency information is the only kind of information required for CP-nets, because there is no relative importance encoded. This implies that the RDF version of the CP-net, in terms of the *full* ontology, would not contain any predicate conditionallyMoreImportantThan and irrespectiveOf or any instance of the class InstanceOfRelativeImportance.

8.4.2 Ordering SPARQL results via CP-theories

In the following, we assume that users are looking for the best k items satisfying some requirements and that the choice for the best ones is led by their preferences, formulated according to a CP-theory Γ, on a set of variables $V = \{X_1,...,X_n\}$. Hence, we aim at solving a top-k query answering problem, where the ordering criterion is encoded in Γ. In the presented approach, we concentrate on cuc-acyclic CP-theories, to preserve the nice computational properties introduced in Section 2.2.1 and exploit the algorithmic approach suggested by Theorem 2.

The ultimate goal of our proposal can be summarized by the following query formulated in a meta-language on top of SPARQL:

SELECT ?item
WHERE {
    ?item **satisfies** *user requirements*
}
ORDER BY Γ
LIMIT k

Here, *user requirements* are represented by a SPARQL graph pattern where at least one triple has ?item as subject. In the following, we use the notation $\mathscr{R}$(?item) to denote the user requirements associated with the variable ?item.

Example 7 (Books cont'd)
"*Giorgio really wants to relax, and so he is looking only for books with more than 300 pages*". In this case, Giorgio's requirements $\mathscr{R}$(?item) are represented by:

```
?item a dbo:Book.
?item dbo:numberOfPages ?page.
FILTER(?page>300).
```
■

The computation of an answer to the previous query, goes through the exploitation of the *full* version of Γ. The overall approach is composed of two main steps.

Step 1. Here, we compute a representation of α and β, starting from each $\varphi \in \Gamma$ and, by exploiting Theorem 2, an ordering based on $\gg_\Gamma$ for all possible outcomes is eventually returned. The representation of α and β as URIs goes through a string concatenation (we use the `GROUP_CONCAT` aggregate function of SPARQL).

Step 2. This step deals with ranking the items matching $\mathscr{R}$(`?item`) according to $\gg_\Gamma$ as computed in the previous step.

Both steps are detailed in the following.

Ordering the Outcomes (Step 1). From Theorem 2 in Section 2.2.1, we know how to build a strict partial order on a set of outcomes $\mathscr{O}$ extending $>_\Gamma$ by comparing outcomes via $\gg_\Gamma$. By means of the same meta-language that we used before, the ordering of outcomes can be done via the following query. There, we see the outcomes are ordered according to a counter representing the number of outcomes that they are able to $\gg_\Gamma$-dominate.

```
SELECT ?outcome-Dominating
       (COUNT(?outcome-dominated) AS ?counter)
WHERE {
  FILTER { ?outcome-Dominating ≫▪ ?outcome-dominated }
}
GROUP BY ?outcome-Dominating
ORDER BY  DESC(?counter)
```

In order to compute values for the two variables `?outcome-Dominating` and `?counter`, the previous query should act on one pair (α, β) per time by checking if $\alpha \gg_{\Gamma} \beta$.

The preference-based reasoning is performed exclusively by means of the `SPARQL` 1.1 query *OrderingQuery* whose generation is detailed in Algorithm 2 of Appendix 8.5.

8.5 Query formulation algorithm for CP-theories

Algorithm 2 has the user's preferences graph $\mathscr{G}_{user}$ as input, that is, the `RDF` version of the user's CP-theory Γ in terms of the *full* ontology and returns the `SPARQL` query able to order outcomes according to $\gg_{\Gamma}$, a strict partial order extending $>_{\Gamma}$ (see Theorem 2). Line 3 computes, for each outcome o, the values (of variables in V) that it is composed of, through the string *Outcome_D_values* built with the for cycle that ends on line 2. The string *Outcome_D_values* contains just the names of variables in V with a suffix *_D*. Line 3 also computes the number of outcomes o' that o dominates according to $\gg_{\Gamma}$, referred to as `?counter`. The counting is made possible by the combination of the `COUNT` in line 3 and the `GROUP BY` in line 18. The `?counter` variable is then used by the `ORDER BY` in line 19 to rank the result set. It is worth to notice that line 3 asks for `URI(?outcome_D)` and not just for `?outcome_D`, since this entity will be employed as the subject of triples at the beginning of **Step 2** described in Section 8.4.2. Given a pair of outcomes (o, o'), lines to 16 are used to identify the set of variables $\Theta^{'}(o,o')$ and among them the set of variables $\{X \in \Theta^{'}(o,o')\colon\ o(X) \succ_{o}^{X} o'(X)\}$. The first nested subquery (lines 5 to 13) considers one pair of outcomes at a time, `?outcome_D` and `?outcome_-d`, where `D` and `d` stand respectively for ***D**ominating* and ***d**ominated.* The for loop of lines until 9 allows to consider first `?outcome_D` and then `?outcome_d`. For each of them, the nested loop of lines to 6 introduces the values corresponding to the variables in V. For each variable $X_i \in V$, Algorithm 2 looks for values `?X`$_{i}$`_-`*y* filtering only elements of the set $\{$*value*(x_{i1}), *value*$(x_{i2})\}$ in the binary case, or $\{$*value*(x_{i1}), ... , *value*$(x_{in})\}$ elsewhere, through the `VALUES` assignment at line 6.

The algorithm requires that a list W of variables is defined from the set of variables V of Γ: the variables that refer to each instance of `Condition` must appear in the same order in W, optionally allowing some recurrence. At lines 7-9, the outcome is explicitly built by concatenating, according to the order imposed by W, the values extracted for various `?X`$_{i}$`_`y together with *attribute(X_i)*, for all members of W. Line 10 is added to verify that the pair of outcomes to compare is made of distinct elements.

At line 11, the patterns and the `FILTER` are used to identify the variables `?V` with different values `?value1` and `?value2` in the outcomes `?out-come_D` and `?outcome_d`, namely the variables in the set $\Delta($`?outcome_D`, `?outcome_d`$)$. As imposed by the couple of `FILTER NOT EXISTS` of lines 12 and 13, these variables `?V` are such that:

- there does not exist any variable `?V2` in the set $\Delta($`?outcome_D`, `?outcome_d`$)$ that is `more-ImportantThan ?V`;
- there does not exist any variable `?V3` in the set $\Delta($`?outcome_D`, `?outcome_d`$)$ that is `condi-tionallyMoreImportantThan ?V` under a `Condition` extended by `?outcome_D`.

In particular, the nested subquery appearing within the `FILTER NOT EXISTS` of line 13 extracts only the instances of `Condition` extended by `?outcome_D`, building the representative strings of the conditional values, concatenating them in `?Concatenated` and checking the inclusion of the string `?Concatenated` in `?outcome_D` through `FILTER` and `CONTAINS`. The pair of `FILTER NOT EX-ISTS` of lines 12 and 13 allows therefore to identify the set of variables $\Theta'($`?outcome_-D`, `?outcome_d`$)$.

The `UNION` of Query 1 and Query 2 is added at line 14. It returns a set of quadruples of the general form $\langle$`?V`, `?ConcatenatedParent`, `?Prefer`, `?Over`$\rangle$, able to order an outcome over another one, locally with respect to `?V`. For each variable `?V` within the set $\Theta'($`?outcome_D`, `?outcome_d`$)$, the `IF` of line 15 verifies if one of the quadruples on variable `?V` can be used to order `?out-come_D` over `?outcome_d` locally with respect to `?V`. In particular, for quadruples with a missing value for `?ConcatenatedParent`, it is sufficient to verify if `?outcome_D` contains the better value of a preference, i.e., `?Prefer`, and `?out-come_d` contains the relative worse value `?Over`. Instead, for quadruples with a bound value for `?ConcatenatedParent`, it must happens that `?outcome_D` contains the value of `?ConcatenatedParent`, as well as `?Prefer` and `?out-come_d` contain the value of `?Over`. If one of the `||` (or) conditions happens, the `BIND` instantiate the value of `?counterBind` to 1, otherwise to 0. The `?coun-terBind` value is summed up across all instantiation of `?V` in Θ'(`?outcome_D`, `?outcome_d`), and it resolves into `?counterV` at line 4. The same line also computes the cardinality of Θ'(`?outcome_D`, `?outcome_d`), namely, the value `?counterVundominated`. The `FILTER` at line 17 verifies if the pair of values `?counterVundomi-nated` and `?counterV` coincides, that is, if `?out-come_D`$(X) \succ^{X}_{\texttt{?outcome\_D}}$ `?outcome_d`(X) for all variables X in Θ'(`?outcome_-D`, `?outcome_d`). In conclusion, if the `FILTER` of line 17 returns `true` then `?outcome_D` dominates `?outcome_d` with respect to $\gg_{\Gamma}$, and its `?counter` value is incremented of a unit. Only distinct dominated outcomes are counted, through the solution modifier `DISTINCT` at line 3.

If we consider the query in Appendix 8.6 resulting from the running example on Giorgio's preferences, we see that the variables involved in CP-statements as well as the corresponding values are encoded in the initial part of the query (line 9-50) and in the `GROUP BY` statement (line 130). The remaining of the query is quite standard and does not depend on the underlying CP-theory Γ. As for the initial part of the query, it contains a number of $2 \cdot |V|$ `VALUES` statements, where we assign all the allowed values to variables in V and a `BIND` statement used to compose the strings representing all possible outcomes. We emphasize that the whole query is automatically generated by Algorithm 2, starting from the *full* version of Γ, and then the process is completely transparent to the user.

8.6 Ordering Query for the Book Example

```
prefix cpt:<http://sisinflab.poliba.it/semanticweb/lod/ontologies/cpt_full.owl#>
prefix dbpedia-owl:<http://dbpedia.org/ontology/>
prefix dbpedia:<http://dbpedia.org/resource/>
prefix g:<http://sisinflab.poliba.it/semanticweb/graphs/>
SELECT  (URI(?outcome_D) AS ?URIOutcome) ?genre_D ?country_D ?subwork_D
?filmVersion_D (COUNT(DISTINCT ?outcome_d) AS ?counter)
WHERE
{
{ SELECT DISTINCT ?outcome_D ?outcome_d ?genre_D ?country_D ?subwork_D
?filmVersion_D (COUNT(DISTINCT ?V) AS ?counterVundominated)
(SUM((?counterBind))AS ?counterV)
WHERE
{
{SELECT DISTINCT ?outcome_D ?outcome_d ?V ?genre_D ?country_D ?subwork_D
?filmVersion_D
WHERE
{
VALUES (?genre_D) {
(dbpedia:Crime_fiction)   (dbpedia:Autobiographical_novel)
}
VALUES (?country_D) {
(dbpedia:France)  (dbpedia:United_Kingdom)
}
VALUES (?subwork_D) {
(cpt:subsequentWorkYes)   (cpt:subsequentWorkNo)
}
VALUES (?filmVersion_D) {
(cpt:filmVersionYes)  (cpt:filmVersionNo)
}
BIND (CONCAT(STR(dbpedia-owl:country),STR(?country_D),
STR(dbpedia-owl:literaryGenre),STR(?genre_D),
STR(dbpedia-owl:subsequentWork),STR(?subwork_D),
STR(dbpedia-owl:filmVersion),STR(?filmVersion_D)) AS ?outcome_D).
VALUES (?genre_d) {
(dbpedia:Crime_fiction)  (dbpedia:Autobiographical_novel)
}
VALUES (?country_d) {
(dbpedia:France)   (dbpedia:United_Kingdom)
}
VALUES (?subwork_d) {
(cpt:subsequentWorkYes)   (cpt:subsequentWorkNo)
}
VALUES (?filmVersion_d) {
(cpt:filmVersionYes)   (cpt:filmVersionNo)
}
```

```
BIND (CONCAT(STR(dbpedia-owl:country),STR(?country_d),
STR(dbpedia-owl::literaryGenre),STR(?genre_d),
STR(dbpedia-owl:subsequentWork),STR(?subwork_d),
STR(dbpedia-owl:filmVersion),STR(?filmVersion_d)) AS ?outcome_d).
FILTER(?outcome_D!=?outcome_d).

?V a cpt:Variable.
?V cpt:variableDomain ?variable1. ?variable1 cpt:value ?value1.
?V cpt:variableDomain ?variable2. ?variable2 cpt:value ?value2.
FILTER (!(?value1=?value2)&& CONTAINS(?outcome_D,STR(?value1))
&& CONTAINS(?outcome_d,STR(?value2))).
FILTER NOT EXISTS{
?V2 cpt:moreImportantThan ?V.
?V2 cpt:variableDomain ?vd1. ?vd1 cpt:value ?v1.
?V2 cpt:variableDomain ?vd2. ?vd2 cpt:value ?v2.
FILTER((!(?v1=?v2))&& CONTAINS(?outcome_D,STR(?v1)) &&
CONTAINS(?outcome_d,STR(?v2))).
}
FILTER NOT EXISTS{
?V3 cpt:conditionallyMoreImportantThan ?instanceOfRelativeImportance.
?instanceOfRelativeImportance cpt:hasCondition ?C.
?instanceOfRelativeImportance cpt:hasLessImportantVariable ?V.
?V3 cpt:variableDomain ?vd13. ?vd13 cpt:value ?v13.
?V3 cpt:variableDomain ?vd23. ?vd23 cpt:value ?v23.
{  SELECT DISTINCT ?C
(GROUP_CONCAT(CONCAT(STR(?attr),STR(?value));separator ="")
AS ?Concatenated) WHERE{
?C cpt:contains ?c.
?c cpt:attribute ?attr;  cpt:value ?value.
}
GROUP BY ?C
}
FILTER(CONTAINS(?outcome_D,?Concatenated)).
FILTER(!(?v13=?v23)&& CONTAINS(?outcome_D,STR(?v13)) &&
CONTAINS(?outcome_d,STR(?v23))).
}
}
}
{ SELECT DISTINCT ?V  ?ConcatenatedParent ?Prefer ?Over {
{  SELECT ?V
(CONCAT(STR(?attrPrefer),STR(?valuePrefer)) AS ?Prefer)
(CONCAT(STR(?attrPrefer),STR(?valueOver)) AS ?Over)  WHERE
{?preference cpt:prefer ?p;
cpt:over ?o.
FILTER NOT EXISTS{?preference cpt:given ?condition.}
?V cpt:variableDomain ?p.
?p cpt:attribute ?attrPrefer;
```

```
cpt:value ?valuePrefer.
?o cpt:value ?valueOver.
}
}
UNION
{  SELECT DISTINCT ?V  ?ConcatenatedParent ?Prefer ?Over  WHERE
{
SELECT DISTINCT  ?condition ?V
(CONCAT(STR(?attrPrefer),STR(?valuePrefer)) AS ?Prefer)
(CONCAT(STR(?attrPrefer),STR(?valueOver)) AS ?Over)
(GROUP_CONCAT(CONCAT(STR(?attr),STR(?value));separator ="")
AS ?ConcatenatedParent) WHERE
{ ?preference cpt:given ?condition.
?preference cpt:prefer ?p;
cpt:over ?o.
?V cpt:variableDomain ?p.
?p cpt:attribute ?attrPrefer;
cpt:value ?valuePrefer.
?o cpt:value ?valueOver.
?condition cpt:contains ?c.
?c cpt:attribute ?attr;
cpt:value ?value.
}
GROUP BY ?condition ?V ?attrPrefer ?valuePrefer ?valueOver
}
}
}
}
BIND(IF(  ( (!BOUND(?ConcatenatedParent) &&
CONTAINS(?outcome_D,?Prefer)&&
CONTAINS(?outcome_d,?Over))
||
(BOUND(?ConcatenatedParent) && ?ConcatenatedParent!="" &&
CONTAINS(?outcome_D,?ConcatenatedParent) &&
CONTAINS(?outcome_D,?Prefer) &&
CONTAINS(?outcome_d,?Over)) )  ,1,0) AS ?counterBind )
}
GROUP BY  ?outcome_D ?genre_D ?country_D ?subwork_D ?filmVersion_D ?outcome_d
}
FILTER(?counterV=?counterVundominated)
}
GROUP BY  ?outcome_D ?genre_D ?country_D ?subwork_D ?filmVersion_D
ORDER BY DESC (?counter)
```

The algorithm takes as input the *full* version of Γ and computes a query able to return a list of outcomes ordered according to `?counter`. In particular, the query returns for each outcome, a numerical score representing its position in the ranking imposed by $\gg_\Gamma$.

For a better clarification, the reasoning procedure under the comparison between the pair (α, β) is summarized in the following:

1. The query computes the set $\Theta^{'}(\alpha, \beta)$ by considering the variables X in the set $\Delta(\alpha, \beta)$ for which there do not exist: (i) a variable $X^{'} \in \Delta(\alpha, \beta)$ linked to X by property `cpt:moreImportant-Than` and (ii) a variable $X^{''} \in \Delta(\alpha, \beta)$ which is `cpt:conditionallyMoreImportant-Than` than X under a condition extended by α.

2. It then counts the number of variables X from the set $\Theta^{'}(\alpha, \beta)$ that let to state $\alpha(X) \succ_{\alpha}^{X} \beta(X)$ and compares it to the cardinality of $\Theta^{'}(\alpha, \beta)$;

3. If the numerical values coincide, which means that for each variable X in $\Theta^{'}(\alpha, \beta)$, $\alpha(X) \succ_{\alpha}^{X} \beta(X)$ holds, then the query concludes that $\alpha \gg_\Gamma \beta$.

In order to get all the information needed to check $\alpha(X) \succ_{\alpha}^{X} \beta(X)$ from the *full* version of Γ, *OrderingQuery* embeds Query 1 and Query 2 reported in the following. They return a set of quadruples $\langle$`?V`, `?ConcatenatedParent`, `?Prefer`, `?Over`$\rangle$, with `?ConcatenatedParent` optionally not instantiated, able to locally order α over β with respect to variable `?V`. Given a variable $X_i \in V$ with $dom(X_i) = \{x_{i1}, x_{i2}, ..., x_{in}\}$, we use the following notation relative to the corresponding instances `cpt:`$\mathtt{x_{i1}}$, `cpt:`$\mathtt{x_{i2}}$,..., `cpt:`$\mathtt{x_{in}}$ of the class `cpt:Value`:

- $value(x_{ij})$ is the `object` of the triple `cpt:`$\mathtt{x_{ij}}$ `cpt:value object`;
- $attribute(x_{ij})$ denotes the `object` of the triple `cpt:`$\mathtt{x_{ij}}$ `cpt:attribute object`;
- we call *representative string* of x_{ij} the concatenation of the two strings represented by $attribute(x_{ij})$ and $value(x_{ij})$ respectively. The combination of $attribute(x_{ij})$ and $value(x_{ij})$ is used to represent x_{ij}, as they uniquely identify a value in the domain of a variable. Indeed, in case we used only $value(x_{ij})$, ambiguous situations could arise when it is used in combination with different attributes.

Finally, for an instance `cpt:c` of the class `cp:Condi-tion`, we call *conditional values* of `cpt:c` all the objects of the triples `cpt:c cpt:contains object`.

Query 1

```
1  SELECT ?V
2  (concat(str(?attrPrefer),str(?valuePrefer)) as ?Prefer)
3  (concat(str(?attrPrefer),str(?valueOver)) as ?Over)
4  WHERE
5  {
6  ?preference cpt:prefer ?p;
7  cpt:over ?o.
8  FILTER NOT EXISTS {?preference cpt:given ?condition.}
9  ?V cpt:variableDomain ?p.
10 ?p cpt:attribute ?attrPrefer;
11 cpt:value ?valuePrefer.
12 ?o cpt:value ?valueOver.
13 }
```

Query 1 processes elements φ of Γ with $u_\varphi = \top$. Within the query, they are represented by the variable `?preference`. The selection is made possible by the `FILTER NOT EXISTS` on the pattern `{?preference cpt:given ?con-dition.}`
(line 8). Considering that the objects of properties `cpt:prefer` and `cpt:over` must be distinct values of the same variable, the query firstly extracts the variable that the preference acts on, i.e., `?V` (line 1 and line 9). Then, it computes the *representative strings*, `?Prefer` and `?Over` (lines 2–3) for the objects `?p` and `?o` of the two triples involving `cpt:prefer` and `cpt:over` (lines 10–12).

Query 2

```
SELECT DISTINCT ?V  ?ConcatenatedParent ?Prefer ?Over  WHERE{
SELECT ?condition ?V
(GROUP_CONCAT(concat(str(?attr),str(?value)); separator="")
as ?ConcatenatedParent)
(concat(str(?attrPrefer),str(?valuePrefer)) as ?Prefer)
(concat(str(?attrPrefer),str(?valueOver)) as ?Over)
WHERE
{
?preference cpt:given ?condition;
cpt:prefer ?p;
cpt:over ?o.
?V cpt:variableDomain ?p.
?p cpt:attribute ?attrPrefer;
cpt:value ?valuePrefer.
?o cpt:value ?valueOver.
?condition cpt:contains ?c.
```

```
17 ?c cpt:attribute ?attr;
18 cpt:value ?value.
19 }
20 GROUP BY ?condition ?V ?attrPrefer ?valuePrefer ?valueOver
21 }
```

Differently from the previous query, Query 2 is used to process statements φ belonging to Γ with $u_\varphi \neq \top$. The selection is made via the pattern {`?preference cpt:given ?condition.`} (line 9). Let us consider first the nested subquery in lines 2–20. For each instance of class `Preference`, such query extracts the variable `?V` that the preference is about (lines 2 and 12) and considers the `cpt:given` condition `?condition` (line 9), extracting its corresponding *conditional values* (line 16). The *representative strings* of such *conditional values* are then computed (lines 17–18) and concatenated at lines 3–4 in `?ConcatenatedParent`, grouping by condition. The variables `?Prefer` and `?Over` are defined similarly to Query 1. The external query is just used to restrict the result set to variables `?V`, `?ConcatenatedParent`, `?Prefer`, `?Over`.

Ordering the Items (Step 2). Given the information on outcomes returned by the *OrderingQuery* at the previous step, on both values of variables and position in the ranking, an external RDF dataset, e.g., `DBpedia`, may be queried, asking for items satisfying the hard constraints ($\mathscr{R}$(`?item`)) imposed by the user and such that, when limiting the attention on variables in V, they match the description of an outcome. Items are then ordered according to the ranking over corresponding outcomes.

We are well aware that the one we propose is just a possible rewriting of a CP-theory in a SPARQL query and other encodings are possible, ever more efficient from a computational perspective. Moreover, we may see that the performance of the overall approach decreases when the size of variable domains grows and, in its current version, the approach is not able to handle continuous domains as for distance and time. Nevertheless, we believe that the proposed approach is a good starting point to reason with preferences in a pure Linked Data environment, as it is a straight implementation of theoretical results coming form previous works [280].

8.6.1 Instantiation of the framework

The procedure to retrieve items ordered according to user's preferences is made up of four phases:

- the *loading* of user's preferences;
- an *insert* to add information about outcomes;
- the execution of a *federated query*;
- the (optional) *dropping* of user's preferences.

First of all, the user's preferences file representing the *full* version of Γ is loaded in the `SPARQL` server through a `LOAD` operation and becomes the user's graph of preferences $\mathscr{G}_{user}$.

Example 8 (Book cont'd)
If the path to the `RDF` file encoding Giorgio's preferences (see Listing 8.1) is generally denoted as *path_to_ ttl_file*, the load operation is executed as follows:

```
prefix g:<http://sisinflab.poliba.it/semanticweb/graphs/>

LOAD path_to_ttl_file INTO GRAPH g:Giorgio_preferences
```

■

The *OrderingQuery* able to order the outcomes according to $\gg_\Gamma$ is then executed. The information returned by the *OrderingQuery* is used to integrate the graph of user preferences $\mathscr{G}_{user}$ with additional triples on outcomes. Specifically, we add information about the score of an outcome and its description. Hence, the following triples are defined for each outcome:

- a triple satisfying the pattern

  `?URIOutcome cpt:hasScore ?score`

- a set of triples instantiating the pattern

  `?URIOutcome cpt:hasValueForX`
  `?ValueForX`

  for every variable X of Γ.

Such information are added to $\mathscr{G}_{user}$ through an `INSERT` query.

Example 9 (Book cont'd)
For the CP-theory $\Gamma_{\mathrm{C-LG-SW-F}}$ with Giorgio's preferences, the `INSERT` operation would behave as follows:

```
prefix cpt:<http://sisinflab.poliba.it/semanticweb/
lod/ontologies/cpt_full.owl#>
prefix dbo:<http://dbpedia.org/ontology/>
prefix db:<http://dbpedia.org/resource/>
prefix g:<http://sisinflab.poliba.it/semanticweb/graphs/>

INSERT { GRAPH g:Giorgio_preferences
{?URIOutcome  cpt:hasScore  ?counter .
?URIOutcome  cpt:hasValueForCountry ?country_D;
cpt:hasValueForLiteraryGenre ?genre_D;
cpt:hasValueForSubsequentWork ?subwork_D;
cpt:hasValueForFilmVersion ?filmVersion_D.
}
}
where { GRAPH g:Giorgio_preferences {

OrderingQuery

}
}
```

Here, *OrderingQuery* denotes the ordering query over outcomes returned by Algorithm 2 in 8.5. The output of Algorithm 2 applied to Listing 8.1 is available in 8.6. ■

The next step is the execution of a federated query, composed by two subqueries. The first subquery retrieves the items satisfying the requirements imposed by the user, i.e., $\mathscr{R}$(`?item`), and for each item it looks for the values of variables in Γ. The retrieval of values grounds on the `VALUES` construct for variables that are `cpt:bound true` and on the combination of `BIND`, `IF` and `EXISTS` otherwise.

The second subquery on the user's preferences graph $\mathscr{G}_{user}$, retrieving for each outcome its score and the variables values. A matching between items and outcomes is hence performed through such values and the items are finally ordered according to the position in the ranking of the relative outcome.

The main reason behind the federation of two (or more) endpoints is that: while the graph containing the RDF version of the user's preferences is encoded in the corresponding document (available at *Preference URI* in Fig. 8.1), all the information about the items that we want to retrieve and rank is encoded in a separate dataset, e.g., `DBpedia`. The main assumption here is that the user's preferences are expressed with respect to a reference dataset/vocabulary, which can queried via a SPARQL endpoint.

Example 10 (Book cont'd)
Suppose that Giorgio is interested in the top-5 list of books matching his hard constraints (see Example 7), ordered according to his preferences encoded in the CP-theory $\Gamma_{\mathrm{C-LG-SW-F}}$ of Table 2.1. The federated query to carry out the searching task would be as follows:

```
prefix cpt:<http://sisinflab.poliba.it/semanticweb/
lod/ontologies/cpt_full.owl#>
prefix dbo:<http://dbpedia.org/ontology/>
prefix db:<http://dbpedia.org/resource/>
prefix g:<http://sisinflab.poliba.it/semanticweb/graphs/>

SELECT ?item_D ?score WHERE {
{SERVICE <http://dbpedia.org/sparql>  {
SELECT DISTINCT ?item_D ?genre_D ?country_D
?subwork_D?filmVersion_D WHERE{
?item_D  a  dbo:Book;
dbo:numberOfPages ?page_D.
FILTER(?page_D>300).
?item_D  dbo:literaryGenre ?genre_D;
dbo:country ?country_D.
VALUES (?genre_D)   {
(db:Crime_fiction)
(db:Autobiographical_novel)
}
VALUES (?country_D) {
(db:France)
(db:United_Kingdom)
}
BIND(IF(EXISTS{?item_D dbo:subsequentWork ?object},
cpt:subsequentWorkYes, cpt:subsequentWorkNo
AS ?subwork_D).
BIND(IF(EXISTS{?item_D dbo:filmVersion ?object},
cpt:filmVersionYes,cpt:filmVersionNo)
AS ?filmVersion_D).
```

```
}
}
}
{graph g:Giorgio_preferences {
SELECT  ?score ?genre_D ?country_D ?subwork_D
?filmVersion_D  WHERE{
?s  cpt:hasScore ?score;
cpt:hasValueForCountry ?country_D;
cpt:hasValueForLiteraryGenre ?genre_D;
cpt:hasValueForSubsequentWork ?subwork_D;
cpt:hasValueForFilmVersion ?filmVersion_D.
}
}
}
}
ORDER BY DESC(?score)
LIMIT 5
```

■

Finally, the graph with the user's preferences $\mathscr{G}_{user}$ can be optionally eliminated with a `DROP` operation.

Please note that all SPARQL queries are executed in a *simple entailment between RDF graphs* on the *full* version of Γ as well as on the external dataset for the federated query. Hence, all the queries are executed under the *RDF Entailment Regime* of SPARQL.

Example 11 (Book cont'd)
The graph related to Giorgio's preferences may be dropped as follows:

```
prefix g:<http://sisinflab.poliba.it/semanticweb/graphs/>

DROP GRAPH  g:Giorgio_preferences
```

■

8.7 Application

We now describe a tool[8] implementing the framework described in previous sections and aimed at supporting the end-user in retrieving a list of semantic resources ordered according to her preferences formulated under the CP-theory formalism.

[8]The tool is available at `http://cptheorysparql.cloudapp.net:10002/`

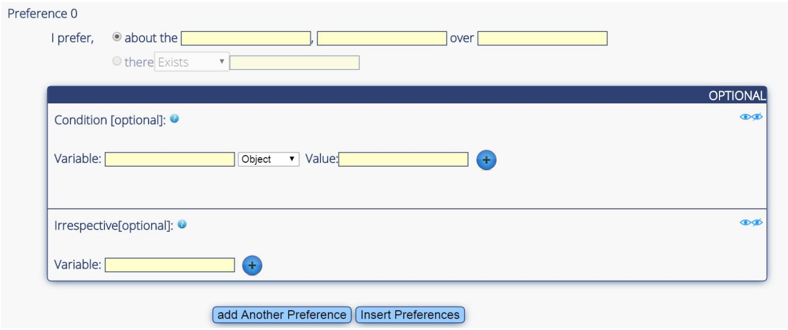


Figure 8.4: Preferences Insertion.

The tool just asks for preferential statements formulated under the CP-theory formalism, i.e., "*given* u_φ, x_φ *is strictly preferred to* x'_φ, *all else being equal, but irrespective of the values of variables in* W_φ". With reference to the ontologies introduced in Section 8.4, this means that for this preliminary step of preference definition, the interested user only has to deal with classes and properties of the *lite* ontology of Figure 8.2. In particular, after the selection of the domain of interest, the user inserts her preferences as depicted in Figure 8.4. The interface manages both instances of variables `cpt:bound true` and `cpt:bound false`, as introduced in Section 8.4. In the former case, the variable which the preference is "about" and the couple of values separated by the word "over" must be introduced; in the latter case, the user has to specify whether the presence or the absence of a variable is preferred. Optionally, she can insert a "Condition" under which the above order holds and make explicit, in the "Irrespective" section, the set of variables for the (conditional) relative importance. She can insert as much preferences as she wants with the "add Another Preference" button or complete the insertion procedure with the "Insert Preferences" button.

When this second button is pressed, the tool takes care that the user has defined the transitive closure of those preferential statements related to multiple values of a variable. More specifically, if $\varphi_1 = u_{\varphi_1} : x > \hat{x}\,[W_{\varphi_1}]$ and $\varphi_2 = u_{\varphi_2} : \hat{x} > \bar{x}\,[W_{\varphi_2}]$ have been inserted, with $x, \hat{x}, \bar{x} \in dom(X)$, and u_{φ_1} and u_{φ_2} do not contain two different values of the same variable, then the rule $\varphi_3 = u_{\varphi_3} : x > \bar{x}\,[W_{\varphi_3}]$ is added, if missing (where u_{φ_3} is the condition joining u_{φ_1} and u_{φ_2}, and W_{φ_3} is the intersection of sets

W_{φ_1} and W_{φ_2}). As an example in the movie domain, one may state that (φ_1) the actor Hugh Grant is preferred over Colin Firth for comedy films irrespective of the country of production and that (φ_2) Colin Firth is preferred over Joaquin Phoenix for movies directed by Woody Allen. In this case, the additional fact to add would be that (φ_3) Hugh Grant is preferred over Joaquin Phoenix for comedy movies directed by Woody Allen ceteris paribus and with no relative importance specification, since the intersection of sets W_{φ_1} and W_{φ_2} is empty. The tool then exploits the *lite* version of Γ to generate its *full* version by managing the transitive closure of `moreImportantThan` and `conditionallyMoreImportantThan`, as described in Section 8.4 [9].

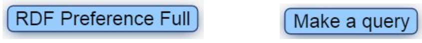
The tool also helps the user to understand if her CP-theory is cuc-acyclic or not. It returns an error message to the user if the CP-theory is not locally consistent or the directed graph $J_{u_\varphi}(\Gamma)$ on V, for any u_φ introduced by the user in her preferential statements, is cyclic. Otherwise, it returns the encoding that can be used to query the `DBpedia` dataset.

As an alternative to the manual insertion of preferences, the user can decide to upload a file of preferences written according to the *lite* ontology, using the specific top right button. The transitivity and cuc-acyclicity checking and the introduction of the additional class and properties of the *full* ontology is performed in this case as well.

At this point, a file with the *full* version of Γ is available and can be visualized by pushing the button in Figure 8.5 (a) or exploited directly to formulate a query against `DBpedia` through the button in Figure 8.5 (b).

More specifically, when the button of Figure 8.5 (b) is pressed, an interface as the one depicted in Figure 8.6 appears. The interface mimics the query presented at the beginning of Section 8.4.2. There, the users may insert their own requirements $\mathscr{R}(\texttt{?item})$ and specify the number k of results to use in the `LIMIT` modifier, which by default is set to 10. The URL displayed after the `ORDER BY` clause represents

[9] The rules that allow the system to manage the definition and the transitive closure of both properties `moreImportantThan` and `conditionallyMoreImportantThan` have been implemented in Prolog and are available at `http://sisinflab.poliba.it/semanticweb/lod/ontologies/rules.pl`.



(a) (b)

Figure 8.5: The buttons to display the Full RDF File (a) or to formulate a preference-based query (b).

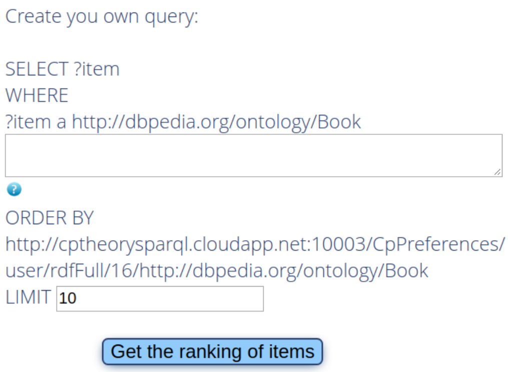


Figure 8.6: The interface to build the query.

the location of the RDF file containing the *full* version of Γ.

The query returns the top-k list of items belonging to the domain of interest (e.g., books as in Figure 8.6), satisfying $\mathscr{R}(\texttt{?item})$ and ordered according to user's preferences.

Example 12 (Book cont'd)
Introducing Giorgio's preferences, contained in the CP-theory of Table 2.1, in the proposed tool would produce the top-5 list of results shown in Table 8.1. The results refer to the release of DBpedia 2015-04[10] (also known as: 2015 A). By looking in DBpedia, one can observe an exact matching with the expected order shown in Example 4, according to the following triples:

```
@prefix db: <http://dbpedia.org/resource/>
@prefix dbo: <http://dbpedia.org/ontology/>

db:An_Uncertain_Place  dbo:country  db:France ;
dbo:literayGenre  db:Crime_fiction .
db:Requiem_for_a_Fish  dbo:country  db:France ;
dbo:literayGenre  db:Crime_fiction .
db:Blood_Red_Rivers    dbo:country  db:France ;
```

[10] http://wiki.dbpedia.org/dbpedia-data-set-2015-04

```
dbo:literayGenre  db:Crime_fiction .
db:Tropic_of_Capricorn_(novel) dbo:country  db:France ;
dbo:literayGenre
db:Autobiographical_novel .
db:Have_Mercy_on_Us_All  dbo:country  db:France ;
dbo:literayGenre  db:Crime_fiction ;
dbo:subsequentWork
db:Wash_This_Blood_Clean_from_My_Hand.
```

■

8.8 Experiments

In order to asses the effectiveness of the presented approach and the implemented tool, we set up two different experiments.

The first experiment consisted of 20 real users using the tool to express their preferences. After the test, users were asked to fill up a questionnaire (reported in Table 8.4). The dataset adopted consisted of a subset of `DBpedia` 2015-04 related to the four popular domains of: Movies, Food, Music and Books. The statistics of the dataset used for experiments are detailed in Table 8.2.

It is worth noticing that CP-statements can also be automatically extracted from users data [168, 170, 152, 167]. Nevertheless, we set up the previous experiment to have a hint on the average number of CP-statements φ needed to model a user profile as well as on what is, from a user perspective, the most tricky version of φ to represent among:

- $\top : x_\varphi > \hat{x}_\varphi[\emptyset]$,
- $u_\varphi : x_\varphi > \hat{x}_\varphi[\emptyset]$,
- $u_\varphi : x_\varphi > \hat{x}_\varphi[W_\varphi]$.

The second experiment consisted of simulating 168 users using the platform and expressing an overall number of 6720 preferences and 6720 queries to retrieve the resources ranked by taking into their preferences. The aim of this experiment was that of evaluating the response time of the overall system in retrieving a list of resources based on a set of user preferences.

8.8.1 Test on Real Users

In order to test the capability of a user to exploit the platform and even to test if human users unaware of CP-theories were able to express their preferences, we selected 20 users that did not know anything about CP-theories and, after a 5 minutes tutorial, we asked them to express their preferences by using our tool. We asked them to insert as many preferences as they wanted for each domain on the platform, and we then asked them to fill up a post-experience questionnaire in order to acquire some feedback about the experience. The motivation of this experiment is twofold: the first information that we wanted to collect was the number of preferences that a user is prone to explicitely express. The result for this evaluation is shown in Table 8.3. The users provided an overall number of 322 preferences. The average numbers per user are quite similar among the different domains (between 4 and 6) with a little higher propensity to express preferences over books w.r.t. songs. The similar average values, and the similar standard deviations, suggest that there exists a commonality in the number of expressed preferences over a specific domain.

The second relevant information that we wanted to collect is how much the CP-theories expressiveness may fit a "natural" way of expressing preferences by a human being. To this aim, we submitted a small questionnaire with 10 questions whose relative answers in aggregate form are shown in Table 8.4 and Fig. 8.7. All the questions but Q.2 needed to express a value in a 5-star rating scale, with 1 being the worst answer and 5 the best one.

Users felt that representing preferences was not a trivial operation (3.2 corresponds to the lowest value of the overall questionnaire), but this perceived difficulty is clearly dependent on the type of preference (it is worth to notice that for every specific kind of preference, the score is higher than the overall score).

Thanks to the survey, we can list in an increasing order of difficulty the different kinds of preferences:

- *"About a property, I prefer a Value over another Value"* corresponding to $\top : x_\varphi > \hat{x}_\varphi[\emptyset]$.
- *"Given a condition, I prefer a Value over another Value"* corresponding to

$u_\varphi : x_\varphi > \hat{x}_\varphi[\emptyset]$.

- *"About a property, I prefer a Value over another Value irrespectively to a property"* corresponding to $u_\varphi : x_\varphi > \hat{x}_\varphi[W_\varphi]$.

Moreover, if we look at the pie chart in Fig. 8.7, it emerges that the most difficult part of the process was to detect the properties (variables V) on which the preferences should be expressed. Another information that we wanted to collect was if the possible difficulty in expressing preferences is stable or it progressively vanishes as the number of expressed preferences increases. Questions 7, 8 and 9 show that the first preference was quite hard to express, but, as the experience goes on, it becomes much easier reaching an average value of 4.1.

The last relevant information that we wanted to collect is how much the expressiveness of CP-theories can correspond to a perceived "natural" way to express preferences. Even here, the result is interesting, because CP-theories are perceived as a quite good way of expressing preferences with a high value of 3.8.

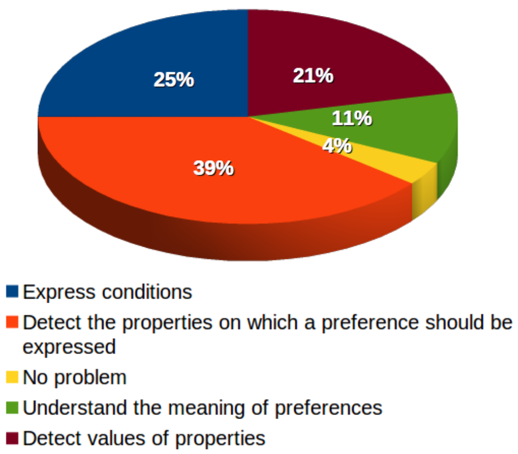


Figure 8.7: Pie chart depicting the results of the second question of the survey.

8.8.2 Test on Simulated Users

In order to closely simulate the behavior of a real user, we designed a tool able to perform the classical operations of expressing a preference and asking the system for an ordered list of relevant resources. For each domain of interest, the simulated users randomly extract (with a uniform distribution) a property that they might be interested in, and then randomly select the other components of the preference (e.g., in case of a simple preference, $\top : x > \hat{x}[\emptyset]$, they select either the more liked resource x and the less liked one $\hat{x}$). The composed preference is then sent to the server to be processed and stored. The system checks if the preference produced a cycle, eventually warning the user (in case of a cycle, a new preference is produced). Once the preference is correctly inserted, the simulated user performs a query to the system to retrieve an ordered list of the 100 most relevant resources. The system continues, inserting a new preference for the same domain, and asking the system for a new list. The process ends when 10 preferences are inserted for each domain and the 10 related queries accomplished. Based on the previous experiment, we considered 10 as a representative number of preferences per user. Fig. 8.8 shows the average execution time for an increasing number of preferences related to the simulated users.[11] The SPARQL engine adopted for the experimental evaluation is Jena Fuseki v. 2.3.1 running on a Linux server (kernel v. 4.4.0-28-generic) with an Intel Xeon @ 2.30GHz CPU and 8 GB RAM, while the local version of DBpedia had been loaded in a Virtuoso Server (v. 07.20.3212), running on a Linux server (kernel v. 4.2.0-23-generic) with an Intel Xeon @ 2.40 GHz CPU and 56 GB RAM.

The results show that queries based on a number of preferences lower than six take approximately less than one second to return results to the user. This is even more interesting if we consider results of the previous experiments, where we saw that users tend to express an average number of preferences between 4 and 6.

8.9 Conclusion and future work

In this study, we have investigated how user preferences can be taken into account while querying `Linked Open Data` datasets. Having realized that the Pareto-

[11] For those interested in a more fine-grained view of the data, a report of the execution times is publicly available at `https://github.com/sisinflab/CP-theories-SPARQL/blob/master/evaluation/evaluationResults.tsv`

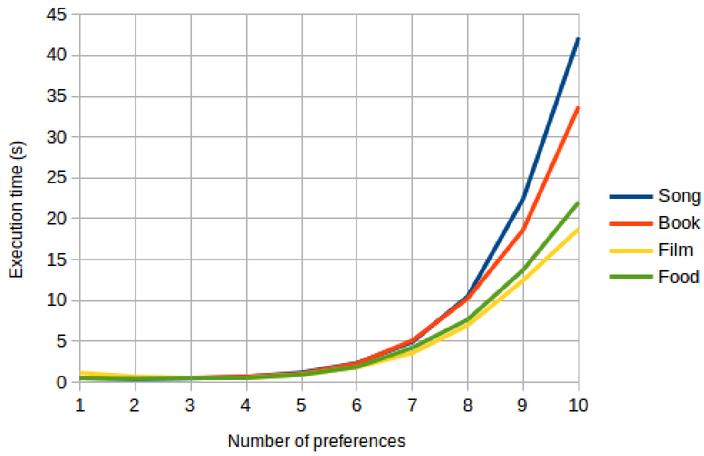


Figure 8.8: Average execution time for increasing number of preferences (1 to 10) in the four domains of: Song, Book, Film and Food.

optimal set identification is not enough, we moved beyond it, proposing an approach to retrieve a ranked list of semantic resources, ordered according to a user's soft constraints. We focused on qualitative preferences, which are closer to how a user makes decisions, especially in a multi-attribute context, and integrated the partial order implied by a qualitative approach into a top-k scenario, that is, returning to the user, who is formulating qualitative preferential statements, a ranked list of resources, optionally limited in its size, ordered according to her preferences. Among qualitative approaches to preference reasoning, we relied on CP-theories, a general and well-known formalism based on the ceteris paribus semantics. We proposed an ontological vocabulary to model CP-theories by means of `RDF` statements under the ceteris paribus semantics. Then, we presented an algorithm able to build a standard `SPARQL` 1.1 query encoding the CP-theory and able to retrieve a ranked set of resources satisfying the corresponding preferential constraints. To our knowledge, this is the first attempt to encode the semantics of a CP-theory into a `SPARQL` query and, along with [226], the first approach that lets `SPARQL` to retrieve a ranked list of resources ordered according to a user's preferences.

We intend the proposed approach as a starting point for many future directions to reason with preferences in a pure Linked Data setting. More efficient encodings for the proposed queries could be investigated, able to mitigate some performance problems, related, for example, to the increase in the size of variables domains. Moreover, in its current version, the algorithm first computes the complete partial order of the outcomes, and then it matches them with items satisfying the users' hard requirements. As the computation of the partial order is computationally expensive, an improvement could surely be to compute and order only those outcomes matching the users' requirements, thus reducing the number of comparisons needed to return query results to the user. As a future direction of our research, we are also working on approaches proposed for automated CP-nets and CP-theories elicitation [77, 110]. Another interesting topic for future research is to explore how our present work can be extended by integrating approaches to preference-based query answering over graph databases, such as the ones in [100, 109, 97], as well as how to deal with variables having continuous domains.

```
procedure GenerateOrderingQueryForCP-Theories(𝒢_user)
    forall X_i ∈ V do
        Outcome_D_values+=?X_i_D
    end
    OrderingQuery = SELECT (URI(?outcome_D) AS ?URIOutcome)
        Outcome_D_values (COUNT(DISTINCT ?outcome_d) AS ?counter)
        WHERE{ ;
    OrderingQuery += {SELECT DISTINCT ?outcome_D ?outcome_d
        Outcome_D_values (count(DISTINCT ?V) AS ?counterVundominated)
        (sum((?counterBind)) AS ?counterV) WHERE{ ;
    OrderingQuery += {SELECT DISTINCT ?outcome_D ?outcome_d ?V
        Outcome_D_values WHERE{ ;
    for y ∈ {D, d} do
        forall X_i ∈ V do
            OrderingQuery += VALUES(?X_i_y) {(value(x_i1)) (value(x_i2))} ;
        end
        OrderingQuery += BIND(CONCAT(STR( ; for i = 1,...,|W| − 1 do
            OrderingQuery += attribute(X_i)), STR(?X_i_y), ;
        end
        OrderingQuery += STR(attribute(X_|W|)), STR(?X_|W|_y)) AS
    ?outcome_y). ;
    end
    OrderingQuery += FILTER(?outcome_D!=?outcome_d).  ;
    OrderingQuery += ?V a cpt:Variable.
        ?V cpt:variableDomain ?variable1.
        ?variable1 cpt:value ?value1.
        ?V cpt:variableDomain ?variable2.
         ?variable2 cpt:value ?value2.
        FILTER (!(?value1=?value2)&&
       contains(?outcome_D,str(?value1))&&
       contains(?outcome_d,str(?value2))).  ;
    OrderingQuery += FILTER NOT EXISTS{    ?V2 cpt:moreImportantThan
  ?V.
        ?V2 cpt:variableDomain ?vd1.
        ?vd1 cpt:value ?v1.
        ?V2 cpt:variableDomain ?vd2.
        ?vd2 cpt:value ?v2.
        FILTER((!(?v1=?v2))&&    contains(?outcome_D,str(?v1)) &&
        contains(?outcome_d,str(?v2))).  } ;
```

Algorithm 2: Algorithm

```
OrderingQuery += FILTER NOT EXISTS{ ?V3
   cpt:conditionallyMoreImportantThan
   ?instanceOfRelativeImportance.
       ?instanceOfRelativeImportance cpt:hasCondition ?C.
       ?instanceOfRelativeImportance cpt:hasLessImportantVariable
   ?V.
       ?V3 cpt:variableDomain ?vd13.  ?vd13 cpt:value ?v13.
       ?V3 cpt:variableDomain ?vd23.  ?vd23 cpt:value ?v23.
       {Select distinct ?C
 (GROUP_CONCAT(CONCAT(str(?attr),str(?value)); separator ="") as
   ?Concatenated)
        where{ ?C cpt:contains ?c.
        ?c cpt:attribute ?attr; cpt:value ?value.  }
        GROUP BY ?C }
   FILTER(CONTAINS(?outcome_D,?Concatenated)).
   FILTER(!(?v13=?v23)&& contains(?outcome_D,str(?v13)) &&
    contains(?outcome_d,str(?v23))).  } }} ;
OrderingQuery += {SELECT DISTINCT ?V ?ConcatenatedParent
       ?Prefer ?Over WHERE{ {Query 1} UNION {Query 2} }} ;
OrderingQuery += BIND( IF( (( !BOUND(?ConcatenatedParent) &&
   contains(?outcome_D,?Prefer)&& contains(?outcome_d,?Over))
       || ( BOUND(?ConcatenatedParent) && ?ConcatenatedParent!=""
   && contains(?outcome_D,?ConcatenatedParent) &&
   contains(?outcome_D,?Prefer) && contains(?outcome_d,?Over)))
       ,1,0) as ?counterBind ) ;
OrderingQuery += } GROUP BY ?outcome_D Outcome_D_values
       ?outcome_d } ;
OrderingQuery += FILTER(?counterV=?counterVundominated)} ;
OrderingQuery += GROUP BY ?outcome_D Outcome_D_values  ;
OrderingQuery += ORDER BY DESC (?counter) ;
return OrderingQuery
```

?item_D	?score
db:An_Uncertain_Place	15
db:Requiem_for_a_Fish	15
db:Blood_Red_Rivers	15
db:Tropic_of_Capricorn_(novel)	13
db:Have_Mercy_on_Us_All	9

Table 8.1: The top-5 list of items retrieved for preferences in the CP-theory of Table 2.1.

Classes	**Instances**	**Properties**	**1 hop resources**
dbo:Book	31172	36	12964
dbo:Film	90063	31	82922
dbo:Food	6003	21	2367
dbo:Song	7195	27	2220
Total	**134433**	**115**	**100473**

Table 8.2: Dataset Statistics

	Total	**Min**	**Max**	**Mean**	**Std Dev**
dbo:Book	109	1	11	5.7368	2.1562
dbo:Film	78	1	17	4.5882	3.8900
dbo:Food	75	1	12	4.1667	2.6844
dbo:Song	60	1	12	3.5294	2.6485

Table 8.3: User Experiments Statistics

Q.N.	Questions	Min	Max	Mean	St Dev
Q.1	*How easy has been to represent your preferences?*	1	5	3.1667	1.1100
Q.2	*Which among these did you consider the hardest?*	See Fig. 8.7			
Q.3	*How easy is to represent a preference like* *"About a property I prefer a Value over another Value"?*	1	5	4.1667	1.3048
Q.4	*How easy is to represent a preference like* *"I prefer a resource that has a certain property"?*	2	5	3.5556	1.0966
Q.5	*How easy is to represent a preference like* *"Given a condition I prefer a Value over another Value"?*	2	5	3.8889	0.9852
Q.6	*How easy is to represent a preference like* *"About a property I prefer a Value over another Value irrespectively to a property"?*	2	5	3.6111	1.2005
Q.7	*How easy was to represent the first preference?*	1	5	3.2222	1.3492
Q.8	*After the first preference how easy was to represent the next two ones?*	2	5	3.9444	0.9852
Q.9	*After the first three preferences how easy was to represent the next ones?*	1	5	4.1111	1.2783
Q.10	*How much this way of expressing preferences is similar to your own?*	1	5	3.7778	1.1448

Table 8.4: User Survey Statistics

Chapter 9

Hybrid Relevance: How to enhance traditional relevance weighting schemes

9.1 Introduction

In the last years, many recommendation approaches have been proposed that take advantage of side information to enhance the performance of latent factor models. Side information can refer to items as well as users [276] and can either be structured [255] or semi-structured [297, 25, 62]. Interestingly, in [296] the authors argue about a new generation of knowledge-aware recommendation engines able to exploit information encoded in *knowledge graphs* (*KG*) to produce meaningful recommendations: "*For example, with knowledge graph about movies, actors, and directors, the system can explain to the user a movie is recommended because he has watched many movies starred by an actor*". Hence, the use of side information could not only be used to improve the recommendations but also the whole user experience, by providing explanations or some kind of reasoning from the system.

Nonetheless, in recommendation scenarios where user interactions with the system is of key importance, we cannot deal only on content/context-aware data, but we do need to take into account also collaborative one. The point here is: how to enhance, in a principled way, knowledge about the relevance of each attribute by also encoding collaborative information?

In this research line, we tackle this problem by drawing conceptual and methodological techniques from *Information Retrieval* (*IR*) and *Recommender Systems* (*RS*) in our collaborative-aware relevance framework `CRel-FM`. We start from a generic relevance measure in *IR* (such as *TF-IDF* [24] or *BM*25 [135, 136, 225]) to formally represent the importance or informativeness of the attributes extracted from user and item descriptions [24]. Then, we make use of *Bayesian Personalized Ranking* (*BPR*) optimization procedure [218] to combine the relevance measurements obtained before with the collaborative patterns extracted from the user-item interactions. This augmented representation of items (or users) attributes can be eventually used to compute better item-item (or user-user) similarity values exploited in recommendation engines where the main goal is to present the best possible (most interesting) items to the user as a *top-N* ranking (as opposed to other works where the task is to predict the rating of a user towards an item). We believe that a better understanding of this hybrid representation of attributes relevance will not only improve the quality of recommendations returned to the user, but it will also open up many other possibilities, such as generating better explanations for users or building better suited item and user (latent) representations, by integrating more complex relevance measures or even other optimization techniques tailored to particular recommendation tasks.

More specifically, we address the following main research question: (**RQ1**) In personalized scenarios, whenever we have content-based and collaborative information, how should we exploit them together in a principled way to measure the relevance of an attribute? If a positive answer is found, we then analyze the following: (**RQ2**) May collaborative information improve the quality of attributes' relevance in terms of accuracy, diversity, and novelty of results in a recommendation setting?

The main contributions thus include:

- A principled approach to integrate content and collaborative information by exploiting the relevance of an attribute.
- Extensive experiments on three real-world datasets using state-of-the-art recommendation algorithms.
- Positive results in terms of accuracy, diversity, and novelty for our approach based on hybrid relevance in a majority of the tested scenarios, evidencing its generalization capabilities and potential to capture user and item preferences in different domains.

The reminder of this chapter is structured as follows: in the next section we introduce the proposed framework `CRel-FM` for collaborative-aware computation of attributes relevance. Then, in Section 9.3 we describe the experimental setting to prove the effectiveness of `CRel-FM` and discuss the obtained results. We continue with a description of the main related works in Section 9.4. Conclusion and future work close the chapter.

9.2 `CRel-FM`: Collaborative-aware relevance for recommendation

We present here `CRel-FM`, a collaborative-aware relevance framework particularly well-suited for recommendation scenarios. It mainly bases on *Factorization Machines* (*FM*) and exploits content-based relevance measures that can be plugged-in the overall framework.

In the following, we will consider a typical recommendation setting where we have two separate sets of users $u \in U$ and items $i \in I$ such that $U \cap I = \emptyset$ and the corresponding user-item interaction matrix containing all the implicit ratings (i.e., interactions) users have given to items. In case we have access to some form of items description, such as keywords, tags, attributes, etc., as well as users' one, for recommendation purposes, we usually exploit them to compute similarity values through a relevance measure ρ. For instance, in case items are endowed with a textual representation we may use $\rho = TF\text{-}IDF$ and associate a relevance value ω to each keyword in the text. Each item i has then a vector-based representation

$$i^{TF\text{-}IDF} = (\omega_0^{TF\text{-}IDF}, \ldots, \omega_{|A|}^{TF\text{-}IDF})$$

with A (for attributes) being the set of all possible keywords. In an analogous way, for the computation of $u^{TF\text{-}IDF}$ we may consider all the items enjoyed by u and, even here, compute the ω values associated to each keyword.

This can be further generalized. Indeed, given a relevance measure ρ and a set of attributes A, we can always represent an item as

$$i^{\rho} = (\omega_0^{\rho}, \ldots, \omega_{|A|}^{\rho})$$

The same can be done with u^{ρ}. In case we want to compute an attribute-based similarity value between items or users, both u^{ρ} and i^{ρ} are then the perfect candidates.

In recommendation scenarios, the adoption of factorization models has shown its effectiveness since its initial introduction [216]. In fact, thanks to their subtle modeling of user-item interactions, such models are very precise and effective even in very sparse settings. *FM* have been proposed as a unifying framework to represent all the different factorization models in a general and theoretically sound framework. Without loss of generality, in the following we describe our proposal of `CRel-FM` by looking at *FM* of order 2 for a recommendation problem involving only implicit ratings. The model can be easily extended to a more expressive representation by taking into account, e.g., demographic and social information [8], multi-criteria [6], and even relations between contexts [301].

For each pair of user u and item i we build the binary vector $\mathbf{x^{ui}} \in \{0,1\}^{1\times(|U|+|I|)}$ representing the interaction between u and i in the original user-item rating matrix. In this modeling, $\mathbf{x^{ui}}$ contains only two 1 values corresponding to u and i while all the other values are set to 0. Based on all possible vectors $\mathbf{x^{ui}}$ we then build the matrix $\mathbf{X} \in \{0,1\}^{(|U|\times|I|)\times(|U|+|I|)}$ containing as rows all possible $\mathbf{x^{ui}}$ we can build starting from the original U-I matrix as shown in Fig. 9.1.

	User					Item					
	U_1	U_2	U_3	U_4	...	I_1	I_2	I_3	I_4	I_5	...
x^1	1	0	0	0	...	1	0	0	0	0	...
x^2	1	0	0	0	...	0	1	0	0	0	...
x^3	1	0	0	0	...	0	0	1	0	0	...
x^4	0	1	0	0	...	0	0	1	0	0	...
x^5	0	1	0	0	...	0	0	0	1	0	...
x^6	0	0	1	0	...	1	0	0	0	0	...
x^7	0	0	1	0	...	0	0	1	0	0	...

Figure 9.1: A visual representation of the interaction matrix **X**.

$$\hat{y}(\mathbf{x^{ui}}) = w_0 + \sum_{j=1}^{|U|+|I|} w_j \cdot x_j + \sum_{j=1}^{|U|+|I|} \sum_{p=j+1}^{|U|+|I|} x_j \cdot x_p \cdot \sum_{f=1}^{k} v_{(j,f)} \cdot v_{(p,f)} \qquad (9.1)$$

The *FM* score $\hat{y}(\mathbf{x^{ui}})$ for each vector $\mathbf{x^{ui}}$ is defined as in Equation (9.1), where the parameters to be learned are, respectively: w_0 representing the global bias; w_j giving the importance to every single x_j; and the pair $v_{(j,f)}$ and $v_{(p,f)}$ in $\sum_{f=1}^{k} v_{(j,f)} \cdot v_{(p,f)}$ measuring the strength of the interaction between each pair of variables: x_j and x_p. The summation $\sum_{f=1}^{k} v_{(j,f)} \cdot v_{(p,f)}$ represents the dot product between two vectors: $\mathbf{v}_j$ and $\mathbf{v}_p$ with a size equal to k. Hence, $\mathbf{v}_j$ represents a latent representation of a user, $\mathbf{v}_p$ that of an item within the same latent space, and their interaction is evaluated through their dot product[1]. The number of latent factors is represented by the hyper-parameter k whose value is usually selected at design time. In order to compactly represent all the vectors $\mathbf{v}_j$ and $\mathbf{v}_p$ we introduce the matrix $V \in \mathbb{R}^{(|U|+|I|)\times k}$ having $\mathbf{v}_j$ in the first $|U|$ rows and $\mathbf{v}_p$ in the last $|I|$ ones. For training purposes, V moves from an initial value V_0 to the final $\hat{V}$ where the values in V_0 can be initialized following different strategies [126]. After the training, the first $|U|$ rows of $\hat{V}$ represent a latent representation of each user u with respect to the k latent features while the last $|I|$ ones are the latent representation of all the items i in the catalog. Such vectors can be eventually used to compute the similarity between items or users which is usually exploited in recommendation algorithms. We may say that each value in $\hat{V}$ represents a collaborative-based relevance value of the corresponding latent attributes of u (first $|U|$ rows) or i (last $|I|$ rows). If we look at the mathematical

[1] In our case, we have $v_{(j,f)} \cdot v_{(p,f)} \neq 0$ only when $x_j = 1$ and $x_p = 1$ at the same time. Hence, the final value represents the strength in the interaction between the corresponding u and i.

formulation of *FM*, we see it is able to encode only collaborative information thus being completely agnostic to the nature of catalog items the user interacts with. As a consequence, the same holds for the final similarity values between users or items.

The main idea behind `CRel-FM` is to combine the attribute-based relevance representation of u^ρ and i^ρ with the collaborative one computed by *FM* in a principled and effective way. As a first step, given a relevance measure ρ and a set of attributes A, for each $i \in I$ and $u \in U$ we compute the corresponding vectors i^ρ and u^ρ. Then, we build the matrix $W \in \mathbb{R}^{(|U|+|I|) \times |A|}$ where the first $|U|$ rows correspond to u^ρ vectors and the last $|I|$ rows to i^ρ vectors. Eventually, we set $k = |W|$ in Equation (9.1) and initialize the corresponding matrix V with $V_0 = W$. In other words, we impose the number of latent factors equal to the number of all the attributes A and we then inject the relevance values ω for both u and i within the factorization model. As a consequence, at the end of model training we have the matrix $\hat{V} \in \mathbb{R}^{(|U|+|I|) \times |A|}$ containing a representation of the original content-based values of u^ρ and i^ρ enhanced with collaborative information coming from the *FM* modeling.

Our intuition is that these collaborative-aware relevance values are by far more representative of users and items with respect to the pure collaborative latent rapresentation in the original formulation of *FM* or the pure content-based representation contained in u^ρ and i^ρ. As an example, in Table 9.1 we refer to some attributes extracted from the knowledge graph `DBpedia`[2] and show an example of values obtained after the training (in the column `CRel-FM`) together with the original *TF-IDF* ones [18, **?**] computed for a movie from the `Yahoo!Movies`[3] dataset. The attributes considered here are the categories associated to a movie coming from the corresponding `Wikipedia` ones. It is interesting to compare the attributes ranking coming from the values of *TF-IDF* and the one coming from `CRel-FM` and to see how this latter looks more meaningful than former one.

[2] `http://dbpedia.org`

[3] `http://research.yahoo.com/Academic_Relations`

CRel-FM	TF-IDF	Attribute	CRel-FM	TF-IDF	Attribute
1.3669	0.2584	Space_adventure_films	1.2434	0.2858	Space_adventure_films
1.1252	0.2730	Films_set_in_the_future	1.0355	0.3020	Films_set_in_the_future
0.9133	0.2355	American_science_fiction_action_films	0.8956	0.2605	American_science_fiction_action_films
0.8485	0.3190	1980s_science_fiction_films	0.8951	0.3451	Android_(robot)_films
0.6529	0.1549	Paramount_Pictures_films	0.7338	0.3105	Time_travel_films
0.5989	0.3468	Midlife_crisis_films	0.6665	0.2701	Film_scores_by_Jerry_Goldsmith
0.5940	0.1797	American_sequel_films	0.6581	0.2205	1990s_action_films
0.5862	0.2661	Film_scores_by_James_Horner	0.6561	0.2279	1990s_science_fiction_films
0.5634	0.2502	Films_shot_in_San_Francisco	0.6118	0.1988	American_sequel_films
0.5583	0.1999	1980s_action_thriller_films	0.5649	0.1713	Paramount_Pictures_films

Table 9.1: Top-10 features computed by `CRel-FM` for the movies `"Star Trek II - The Wrath of Khan"` and `"Star Trek - First Contact"`.

9.3 Experimental Evaluation

In this section, we aim at assessing if there is empirical evidence on the usefulness of adopting a hybrid relevance measure to feed recommender systems.

Datasets. To provide an answer to research questions posed in Section 9.1, we have evaluated the performance of our method on three well-known datasets for recommender systems belonging to different domains. The `Last.fm` dataset [53] corresponds to user-artist plays on `Last.fm` online music system released during *HETRec* 2011[4] *Workshop*. It contains social networking, tagging, and music artists listening information from a set of 2*K* users. `LibraryThing` represents books' ratings collected in the `LibraryThing` website[5] community. It contains social networking, tagging and rating information on a [1..10] scale. `Yahoo!Movies` (Yahoo! Webscope dataset ydata-ymovies-user-movie-ratings-content-v1_0)[6] contains movies ratings generated on `Yahoo! Movies` up to November 2003. It provides content, demographic and ratings information on a [1..5] scale, and mappings to `MovieLens` and `EachMovie` datasets. **Feature Selection**. In order to get attributes related to items in the datasets we exploited the freely available mapping[7] that links each item to an entity in the `DBpedia` knowledge graph. Our

[4] `http://ir.ii.uam.es/hetrec2011/`
[5] `https://www.librarything.com/`
[6] `http://research.yahoo.com/Academic_Relations`
[7] `https://github.com/sisinflab/LinkedDatasets`

assumption is that the usage of such well-curated features does not introduce any informative bias (both positive or negative) in the values computed by `CRel-FM` and the eventual recommendations. Following [189], and [208] we filtered out some irrelevant features with a unique threshold for missing values (corresponding to *tm* [189], and *p* [208]). Datasets statistics are shown in Table 9.2.

Dataset	Threshold	#Users	#Items	#Transactions	#Features	Sparsity
Last FM	99.86	1,375	7,312	46,982	1,315	99.53%
`LibraryThing`	99.91	7,221	10,605	313,069	1,169	99.59%
Yahoo! Movies	99.60	4,000	2,528	55,711	747	99.45%

Table 9.2: Datasets statistics.

Experimental Setting. *"All Unrated Items"* [253, 32] protocol has been adopted to compare different algorithms. We have split the dataset using Hold-Out 80-20 retaining for every user the 80% of their ratings in the training set and the remaining 20% in the test set. Moreover, a temporal split has been performed [112] whenever timestamps associated to every transaction is available.

As expressed in our research questions, we want to check if the adoption of our hybrid relevance measure `CRel-FM` is beneficial or not in a recommendation scenario. For this reason each algorithm has been fed using: **i)** the original information contained in *R*; **ii)** the relevance values as computed adopting a relevance measure; **iii)** the new relevance information as defined in Section 9.2. As for point **ii)** we have used *TF-IDF* since we wanted to start with the simplest relevance measure possible.

Algorithms. We evaluate three different families of algorithms that could be fed with relevance information: two *Neighborhood-based* algorithms (`ItemKNN` [236, 237] and `UserKNN` [46]), *Factorization Machines* as a representative of latent factors models (`BPR-FM` [218, 216]), and *Vector Space Model* as a representative of content-based recommender systems (`VSM` [191]). Additionally, we have compared against two non-personalized baselines, i.e., `Random` and `MostPopular`. As for this latter, it is acknowledged that popularity ranking typically shows very good performance because of statistical biases in the data [32] and it is an important baseline to compare against [69].

Table 9.3: Results for `Yahoo!Movies`.

Recommender	Source	P	R	nDCG	EPC	Gini	SE	IC	UC
Random	R	0.001	0.004	0.003	0.001	†**0.826**	†**11.3**	†**2,528**	4,000
MostPopular	R	**0.015**	**0.037**	**0.027**	**0.015**	0.004	3.9	48	4,000
ItemkNN	R	0.040	0.169	0.109	0.042	0.172	9.2	1,808	3,998
	V_0	0.043	0.151	0.119	0.054	**0.353**	**10.0**	**2,456**	4,000
	$\hat{V}$	†**0.055**	**0.197**	†**0.151**	†**0.067**	0.300	9.7	2,405	4,000
UserkNN	R	0.031	0.134	0.086	0.033	0.041	6.8	736	3,998
	V_0	0.030	0.140	0.092	0.033	0.055	7.4	809	4,000
	$\hat{V}$	**0.037**	**0.149**	**0.098**	**0.041**	**0.083**	**7.9**	**1,062**	4,000
FM	R	0.027	0.076	0.050	0.026	0.008	4.7	276	4,000
	V_0	0.039	0.140	**0.113**	**0.050**	**0.289**	**9.8**	**2,320**	4,000
	$\hat{V}$	**0.041**	**0.142**	0.088	0.040	0.021	5.6	594	4,000
VSM	V_0	0.039	0.140	0.113	0.050	0.289	9.8	2,320	4,000
	$\hat{V}$	**0.052**	†**0.198**	**0.149**	**0.063**	**0.297**	**9.8**	**2,357**	4,000

For all the considered recommendation engines we have performed a grid search to tune the hyperparameters. We considered the range of values as suggested by the original authors or by varying the parameters values around the ones showed in the original papers as the best performing ones.

Metrics. In order to evaluate the algorithms, we have measured accuracy through *Precision@N* (*P@N*), *Recall@N* (*R@N*) and *normalized Discounted Cumulative Gain* (*nDCG@N*). *EPC* (*Expected Popularity Complement*) [59] is used to measure novelty, or more precisely the ability of a system to recommend relevant long-tail items. Finally, diversity has been measured through *Item Coverage* (*aggregate diversity* in *top-N* list, *IC@N*), *Gini Index* (*Gini@N*) and *Shannon entropy* (*SE@N*). To measure the ability of producing recommendation lists for each user, *User Coverage* (*UC*) is also computed. The evaluation has been performed considering Top-10 [69] recommendations for all the datasets. A *Threshold-based relevant items* condition [52] of 4/5 has been set for `Yahoo!Movies` and 8/10 for `LibraryThing`, and `Last.fm` respectively in order to take into account only relevant items.

Table 9.4: Results for `Last.fm`.

Recommender	Source	P	R	nDCG	EPC	Gini	SE	IC	UC
Random	R	0.001	0.002	0.001	0.001	†**0.547**	†**12.4**	†**6,395**	1,375
MostPopular	R	**0.025**	**0.064**	**0.035**	**0.023**	0.001	3.7	35	1,375
ItemkNN	R	0.031	0.069	0.042	0.032	0.154	10.5	3,235	1,375
	V_0	0.022	0.039	0.029	0.022	**0.265**	**11.3**	**4,487**	1,373
	$\hat{V}$	**0.037**	**0.070**	**0.051**	**0.041**	0.216	11.0	4,044	1,370
UserkNN	R	0.052	0.122	0.084	0.057	**0.016**	7.2	**831**	1,375
	V_0	0.054	0.130	0.086	0.059	0.012	7.0	420	1,375
	$\hat{V}$	**0.060**	**0.136**	**0.093**	**0.066**	0.015	**7.4**	517	1,375
FM	R	0.042	0.091	0.065	0.046	0.004	5.0	313	1,375
	V_0	0.031	0.055	0.038	0.032	**0.188**	**10.8**	**3,660**	1,375
	$\hat{V}$	†**0.064**	†**0.138**	†**0.099**	†**0.070**	0.020	7.2	897	1,375
VSM	V_0	0.031	0.055	0.038	0.032	**0.188**	**10.8**	3,660	1,375
	$\hat{V}$	**0.050**	**0.087**	**0.062**	**0.053**	0.187	10.7	**3,691**	1,375

9.3.1 Results

Results in Tables 9.3-9.5 show the performance of the different recommender systems fed with different sources. We have marked with **bold** and † the best and the second-best algorithm. The tables correspond respectively to the experiments conducted on `Yahoo!Movies`, `Last.fm`, and `LibraryThing` datasets. To answer RQ2, we analyzed the behavior of the recommender systems on a per dataset basis, focusing on accuracy, diversity, and novelty. While this section is devoted to highlighting the more interesting results, Section 9.3.2 is devoted to drawing some considerations based on the findings depicted here.

`Yahoo!Movies`. As expected, the *Random* recommender shows the highest dataset values in terms of *Gini index*, *Shannon entropy*, and *Item Coverage*. It is worth noticing that *Item-kNN* almost reaches the same performance. *Item-kNN* also shows the best results, at a dataset level, considering *Precision* and *nDCG*. In this case, the behavior of the recommender varying its knowledge source is very clear. Feeding it with V_0, and then $\hat{V}$, we note progressive increases of *Precision*, *Recall*, *nDCG*, and *EPC*. Interestingly, from V_0 to $\hat{V}$ the diversity decreases, whereas the novelty keeps increasing. This is a signal that when we use *R* there is still room for improving all

Table 9.5: Results for `LibraryThing`.

Recommender	Source	P	R	nDCG	EPC	Gini	SE	IC	UC
Random	R	0.001	0.001	0.001	0.001	†**0.774**	†**13.4**	†**11,560**	7,223
MostPopular	R	**0.006**	**0.006**	**0.006**	**0.005**	0.001	3.8	34	7,223
ItemkNN	R	†**0.080**	†**0.146**	†**0.124**	†**0.093**	0.147	10.9	6,378	7,221
	V_0	0.047	0.077	0.059	0.051	0.217	11.4	8,009	7,221
	$\hat{V}$	0.076	0.134	0.112	0.089	**0.245**	**11.7**	**8,258**	7,221
UserkNN	R	0.033	0.070	0.059	0.038	0.022	7.8	1,994	7,221
	V_0	0.035	**0.076**	**0.061**	**0.039**	0.040	8.8	2,489	7,221
	$\hat{V}$	**0.035**	0.071	0.056	0.038	**0.044**	**9.0**	**2,603**	7,221
FM	R	0.024	0.038	0.031	0.023	0.007	6.2	1,122	7,221
	V_0	0.037	0.065	0.047	0.039	**0.211**	**11.4**	**7,437**	7,221
	$\hat{V}$	**0.044**	**0.077**	**0.065**	**0.047**	0.033	8.0	2,687	7,221
VSM	V_0	0.037	0.065	0.047	0.039	0.211	11.4	7,434	7,221
	$\hat{V}$	**0.052**	**0.086**	**0.072**	**0.060**	**0.265**	**11.9**	**8,012**	7,221

the metrics. Instead, when we use $\hat{V}$, we increase accuracy exploiting more items that come from the long tail. Regarding accuracy, the behavior of *User-kNN* is similar, with the same progressive increases. On the other side, the different mechanics of the user-based schema are reflected on an increase in diversity when using $\hat{V}$. *FM* shows different behavior. *Precision* and *Recall* progressively increase changing the knowledge source. *nDCG*, novelty, and diversity show an increase using V_0, but they decrease when $\hat{V}$ is exploited. A possible reason for that is the low number of mapped items. These items are hence described by a low number of features. After the training, these features are over-trained and they affect the performance. This behavior is not present in *k-NNs* algorithms and *VSM*. The *k-NNs* approaches exploit the Neighborhood to alleviate this effect. On the other hand, *VSM* alleviates it by estimating scores based on vector similarity. Since vector are composed of all the features, this effect is absent.

`Last.fm`. It is a completely different dataset, that contains a lesser number of users, and a higher number of items. Moreover, the average number of features that describe an item is 6.6, against the 12.1 of `Yahoo!Movies`. These characteristics affect the behavior of the recommenders. *Random* recommender still shows the highest values for diversity. Differently from the previous experiment, only *Item-kNN* and *VSM* show a good accuracy-diversity trade-off. In *User-kNN* we may notice that are still present the progressive increases for all the accuracy and novelty metrics, but with a vary small margin. *Item-kNN* and *FM* show an interesting commonality. For both recommenders, the exploitation of V_0 produces the negative effect of decreasing accuracy. This may be due to two reasons. First, we have noticed the limited number of features that describe items. Second, maybe those features are not sized for a recommandation task. Nevertheless, after the training, both recommenders show important improvements. Finally, varying the knowledge source in *VSM* leads to remarkable improvements.

`LibraryThing`. This dataset shows a different behavior. In `LibraryThing`, we curiously note that the best recommender is *Item-kNN* fed with *R* matrix. This is definitely due to the small number of features per item: 3.8. However, let us focus on what happens when feeding it with $\hat{V}$. We may notice that accuracy decreases a bit, but the diversity values become much more interesting. Regarding accuracy, *FM* and *VSM* show the same improvements depicted before, with some differences concerning diversity. In *VSM* the item coverage keeps increasing, while *FM* sacrifices diversity in favor of accuracy. The only recommender which shows a confusing behavior is *User-kNN*, in which we may notice an increase of *Precision* and diversity but a decrease of *Recall* and *nDCG*.

9.3.2 Discussion

Once we have detailed the results of the experiments, we can focus on the general findings. If we analyze the behavior of the recommender over the different datasets, we may notice that the exploitation of $TF\text{-}IDF$ relevance is useful for `Yahoo!Movies` and `LibraryThing`. In those datasets, the extra-knowledge leads to better-tuned recommenders. However, this effect is absent in `Last.fm`, where the $TF\text{-}IDF$ is not able to overpass the *R*-based models. Since it happens only in this dataset, in our opinion the reasons for the behavior have to be found in the characteristics of the dataset and a poor description of items. On the other side,

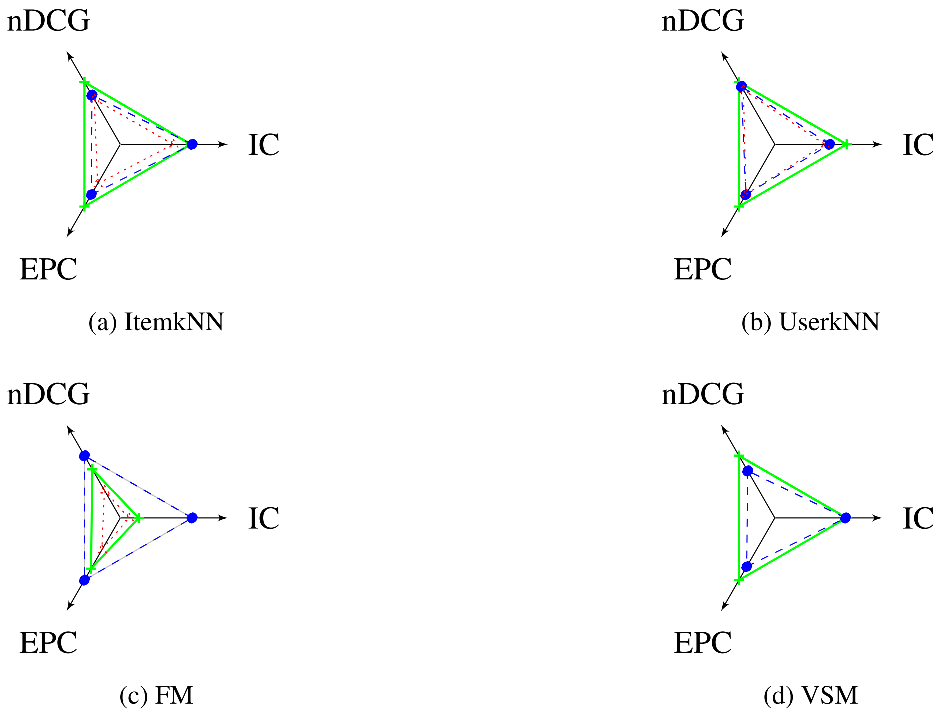


Figure 9.2: Graphical representation of Accuracy, Diversity, and Novelty results for `Yahoo!Movies`. The different knowledge sources, R, V, and $\hat{V}$ are represented respectively through dotted red line, dashed blue line, and solid green line.

the proposed hybrid measure for relevance in almost all cases leads to improvements in accuracy, novelty, and diversity. The training of the features effectively modify the original relevance weights also taking into account a signal of the popularity of the features. Moreover, another trend deserves attention. *Item-kNN* and *FM* show higher values for *Item Coverage* with V_0. This value typically decreases with $\hat{V}$, while accuracy increases. Irrespectively to possible accuracy increases, we observe that *TF-IDF* introduces a noise we consider responsible for those values. After the training phase, we observe a reduction of the recommended items because the noise is partially removed. Some other considerations concern the comparison with [21]. The authors have also considered `Yahoo!Movies` dataset in their analysis, thus we consider mandatory to compare our findings against it. In particular, among the settings they consider, the most similar is `Categorical Setting`. Their `kaHFM` corresponds in our experiments to *Item-kNN* with $\hat{V}$, while *Attribute-based*

Item-kNN corresponds to *Item-kNN* with V_0. In both experiments, we note that *Item-kNN* with $\hat{V}$ is the best performing one, while *Item-kNN*, *Item-kNN* with V_0 and *VSM* show good performance. Since their work is limited to *Precision* and *nDCG*, we can not compare the other dimensions of the analysis. Moreover, since we have considered many different applications of hybrid relevance, further comparison would be difficult. However, the results for `Yahoo!Movies` can be analyzed considering a trade-off perspective. In Fig. 9.2, we may see the four considered models represented on three axes. The idea here is to observe the different trade-offs in terms of accuracy (*nDCG*), diversity (*Item Coverage*), and novelty (*EPC*). The models fed with $\hat{V}$ are denoted by a solid green line. The V_0 models are represented with a dashed blue line and *R* models with a dotted red line. It is clear that for *Item-kNN*, *User-kNN*, and *VSM* there is no trade-off. The $\hat{V}$ model overpasses the other along each dimension. Only *FM* shows different behavior, with $\hat{V}$ shape which is containing *R* shape, but it is contained by the V_0 model. However, here it is clear that the $\hat{V}$ accuracy values are similar to V_0, while the *Item Coverage* is particularly low.

9.4 Related Work

In recent years, some works have proposed to measure relevance exploiting jointly the content information and the collaborative information. In [161], the authors face some known issues of Collaborative-Filtering approaches like *sparsity* and *Cold-Start*. They propose a Bayesian generative model called *Collaborative Variational AutoEncoder* (*CVAE*). *CVAE* learns implicit relationships between items and users from both content and rating information. In their experiments, they show that *CVAE* outperforms the *Collaborative Topic Regression model* (*CTR*), the *Collaborative Deep Learning model* (*CDL*), and the *Deep Music* neural model. They point out that the generative nature of the method eases to capture the key points of items and users. In a subsequent work [118], the authors propose another Deep Neural Network architecture to consider jointly side information and rating information. In detail, they propose *CAVAE*, a *Collaborative Additional Variational AutoEncoder*, in which content (tag information) feeds an additional variational autoencoder. They show that their Bayesian probabilistic model outperforms the *CTR* model, the *CDL* model, and *CVAE*. In *entity2rec* [199], the authors consider a knowledge graph that models $user-item$ and $item-item$ relations. They exploit it to learn property-specific latent representations. From these representations, the authors compute $user-item$ relatedness features to feed two Learning to Rank algorithms (*Adarank* and *LambdaMart*). In [84] the authors combine a *bag-of-words* (*BOW*) representation with a *Convolutional Neural Network* (*CNN*) to create an overall model to retrieve semantically similar questions. Since each original question is a transaction, the collaborative information is naturally injected during the training phase.

An interesting work is [43], in which the authors summarize the contributions of the *Social Information Retrieval* community. Their work is particularly interesting for us because they explicitly exploit collaborative information to improve Information Retrieval techniques. The main difference is that they exploit the $user-user$ connections in a social graph which are a completely different signal from $user-item$ transactions. In a different scenario, the social tagging task, a specific research line focuses on exploiting personal tagging preferences. Even then, relevance information is affected by user behavior. Relevant work is [238] where the authors compute local interaction networks among users exploiting comments. Instead, in [160], the tagging preferences are represented through tag statistics. More recently, in [165], the authors learn a user-specific embedding space. Collaborative information is also exploited in [234], where the authors propose a framework that takes into account user affinities. Interestingly, in [214], the goal is discovering latent associations between users, images, and tags. Another work that focuses on modifying tag relevance using user information is [162], in which two simple methods are proposed to update relevance: *Borda Count* and *UniformTagger*. The latter, proposed by the authors, exploits a neighbor voting algorithm to combine multiple tag relevance estimations.

Another relevant research line is Relevance Feedback Environment, in which feedback information is exploited to correct relevance values. This line of investigation is not closely related to ours. Nevertheless, we have decided to mention it because, even here, the content information is combined with user transactions. Relevance feedback is a well-established technique for improving Interactive Information Retrieval. We can find a specific reference to feature relevance since 1994, in [49], where authors have proposed a modified Rocchio's relevance feedback approach to improve accuracy. This idea is then reflected in [282], in which the authors propose an *idealized relevance feedback* (*IRF*) framework. To further improve the previous work, they perform a term normalization and selection based on the ranking position.

`CRel-FM` is also related to the exploitation of explicit features in Matrix Factorization. One of the first models of this kind is *Explicit Factor Model* (*EFM*) [297]. In this model, products' features and users' opinions are extracted with phrase-level sentiment analysis from users' reviews to feed a matrix factorization framework. Since then, a few improvements to *EFM* have been proposed to deal with temporal dynamics [298] and to use tensor factorization [62]. In particular, in the latter the aim is to predict both user preferences on features (extracted from textual reviews) and items. This is achieved by exploiting the *Bayesian Personalized Ranking* (*BPR*) criterion [218].

In `CRel-FM`, we exploit the *Factorization Machines* (*FM*) data model [216]. FMs are the most widely used factorization models because they offer a number of advantages compared against other latent factors models such as *SVD++* [148], *PITF* [221], *FPMC* [219]. The main advantage is that *FMs* are designed for a generic prediction task. On the other side, other mentioned techniques are task-specific. Moreover, *FM* is a linear model where parameters can be estimated accurately even in a high data sparsity scenario. Nevertheless, many researchers have proposed variants of *FMs*. In [120], the authors propose *Neural Factorization Machines* to increase the expressiveness of *FM*. In particular, they enable *FMs* to capture the non-linear structure of real-world data exploiting the non-linearity of neural networks. Furthermore, *Attentional Factorization Machines* [285] use an attention network to learn the relevance of feature interactions. Finally, in [220], the authors propose a specialized Context-Aware variant of *FMs*.

In a *top-N* recommendation task, recommenders provide users a shortlist of recommendations. For this reason, correctly sorting recommendations has gradually replaced the rating prediction task [172]. These *Learning to Rank* [57] algorithms can be further categorized in *Point-wise* [151], *Pair-wise* [218, 169] and *List-wise* [245, 244]. In particular, *Pair-wise* approaches are usually considered as a good trade-off between ordering performance and computational complexity. Among this class of algorithms, *Bayesian Personalized Ranking* (*BPR*) [218] is one of the most widely adopted. It is based on a simple stochastic gradient descent algorithm to learn the relative order between positive items (items that a user has experienced in his past history) and negative items (items never rated by the user). *BPR* can be easily applied to *Matrix Factorization* and *Factorization Machines* (as in `CRel-FM` and in [26]).

9.5 Conclusion and Future Work

In this chapter, we propose and test `CRel-FM`, a principled method to compute the relevance of item attributes. Its crucial insight is that of exploiting collaborative information in a recommendation scenario to fine-tune the relevance of users/items attributes. The overall approach goes through the exploitation of Factorization Machines to map latent factors to actual attributes and the training of the model adopting a *Learning to Rank* optimization criterion to inject collaborative information. Through an extensive experimental evaluation, we have found that the proposed notion of hybrid relevance provides significant improvements to Recommender Systems even when we adopt the very simple *TF-IDF* relevance measure combined with collaborative information via `CRel-FM`. Furthermore, the exploitation of different signals has led to more accurate and diverse recommendations proving that this could be a solution to the classical accuracy-diversity trade-off [269]. In this work, we have shown the effectiveness of the approach by relying only on one traditional relevance measure (*TF-IDF*). As for our future work, we plan to investigate the behavior of `CRel-FM` with *Entropy-based* [275], *BM*25 [135, 136], and *BM*25*F* [135, 136, 225] weighting schemes. Another investigation direction would be that of replacing the *BPR* optimization with other *pair-wise* and *list-wise Learning to*

Rank optimization algorithms like *AdaRank* [286] and *LambdaMart* [284]. Moreover, we plan to evaluate the method in a more general Information Retrieval setting and to enable explanation services.

Part III

Conclusion and Future Work

Before continuing, some warnings are required. This short final chapter has a quite different tone. Most of this research thesis is written in the first plural person. I have chosen this way because I know that all my achievements are not only mines. My supervisors and all the researchers I collaborate with have contributed. Without their effort, my ideas, my experiments, would still be rough. However, this chapter is devoted to my overall conclusions of this research work. For this reason, if you are looking for the results of the research lines, I am sad to say that you are in the wrong place. I have written each specific research line section to be self-conclusive. Therefore, these lines will be devoted to a broader vision of this research work. In these years, we have analyzed the impact of semantic knowledge on Recommender Systems. We have analyzed the different ways to inject semantic knowledge in the best recommenders. We have proposed several new factorization techniques to factorize user-feature matrices, fix the latent factors' meaning, and compute the best and the worst factors. We have written SPARQL queries able to exploit preference theories to produce recommendations. We have proposed new high-performance interpretable models. Then, we also have explored the world of semi-structured knowledge. In this sense, we have analyzed the role of Time and the Popularity of items. We have exploited Time to build a new diversification algorithm and to define a new personalized popularity algorithm. We have investigated the notion of similarity, and we have proposed a new family of similarities. We have explored the evaluation of Recommender Systems, proposing a new generalized fairness metric and a technique to analyze the discriminative power of evaluation metrics during the training phase. All the results of the above approaches are competitive or better than the state-of-the-art ones. All our findings have been peer-reviewed, and some extensions are still unpublished. There is still room for improvement for each of these techniques. I have indicated some of these possible extensions at the end of each research line. However, some of the most important research pieces of this research path are not in this work. Failed experiments, wrong settings, and imaginative theories have marked this research path. Without those failures, achievements would not be there. This research path, intended as defeats and victories, has deeply changed me. My point of view is changed. In the beginning, the idea of obtaining bad results terrified me. Now I am proud of my failures because I have learned to analyze them to understand them. I have learned to be very rigorous, preparing experiments and formalizing a new theory. When I see the results, I have learned to look further ahead. More important, I have learned that I have many limits. Comparing against research giants forces you to analyze yourself. As for the proposals you have read, also for me, there is room for improvement. Writing style, abstracting at a very high level, connecting theoretical points are only some of the aspects I will work on. In my humble opinion, my research work is only a small rock of a mountain that could be built exploiting knowledge representation and machine learning together. Interpretability of black-boxes, evaluation of fairness, automatic completion of knowledge bases are only some of the topics these technologies can face. On the other side, the knowledge representation background has been crucial also for dealing with semi-structured knowledge, proposing new time-aware and dissimilarity-based algorithms.

There is still much to discover, and failures to understand.

Bibliography

[1] B. Abdollahi and O. Nasraoui. Explainable matrix factorization for collaborative filtering. In J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks, and B. Y. Zhao, editors, *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume*, pages 5–6. ACM, 2016.

[2] B. Abdollahi and O. Nasraoui. Explainable restricted boltzmann machines for collaborative filtering. *CoRR*, abs/1606.07129, 2016.

[3] F. Abel, S. Araújo, Q. Gao, and G. Houben. Analyzing cross-system user modeling on the social web. In S. Auer, O. Díaz, and G. A. Papadopoulos, editors, *Web Engineering - 11th International Conference, ICWE 2011, Paphos, Cyprus, June 20-24, 2011*, volume 6757 of *Lecture Notes in Computer Science*, pages 28–43. Springer, 2011.

[4] F. Abel, E. Herder, G. Houben, N. Henze, and D. Krause. Cross-system user modeling and personalization on the social web. *User Model. User-Adapt. Interact.*, 23(2-3):169–209, 2013.

[5] G. Adomavicius and Y. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Trans. Knowl. Data Eng.*, 24(5):896–911, 2012.

[6] G. Adomavicius and Y. Kwon. Multi-criteria recommender systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 847–880. Springer, 2015.

[7] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6):734–749, 2005.

[8] G. Adomavicius and A. Tuzhilin. Context-aware recommender systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 191–226. Springer, 2015.

[9] V. W. Anelli, P. Basile, D. G. Bridge, T. D. Noia, P. Lops, C. Musto, F. Narducci, and M. Zanker. Knowledge-aware and conversational recommender systems. In S. Pera, M. D. Ekstrand, X. Amatriain, and J. O'Donovan, editors, *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 521–522. ACM, 2018.

[10] V. W. Anelli, V. Bellini, A. Calì, G. D. Santis, T. D. Noia, and E. D. Sciascio. Querying deep web data sources as linked data. In R. Akerkar, A. Cuzzocrea, J. Cao, and M. Hacid, editors, *Proceedings of the 7th International Conference on Web Intelligence, Mining and Semantics, WIMS 2017, Amantea, Italy, June 19-22, 2017*, pages 32:1–32:7. ACM, 2017.

[11] V. W. Anelli, V. Bellini, T. D. Noia, W. L. Bruna, P. Tomeo, and E. D. Sciascio. An analysis on time- and session-aware diversification in recommender systems. In M. Bieliková, E. Herder, F. Cena, and M. C. Desmarais, editors, *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, UMAP 2017, Bratislava, Slovakia, July 09 - 12, 2017*, pages 270–274. ACM, 2017.

[12] V. W. Anelli, V. Bellini, T. D. Noia, and E. D. Sciascio. Knowledge-aware interpretable recommender systems. In I. Tiddi, F. Lécué, and P. Hitzler, editors, *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*, volume 47 of *Studies on the Semantic Web*, pages 101–124. IOS Press, 2020.

[13] V. W. Anelli, A. Calì, T. D. Noia, M. Palmonari, and A. Ragone. Exposing open street map in the linked data cloud. In H. Fujita, M. Ali, A. Selamat, J. Sasaki, and M. Kurematsu, editors, *Trends in Applied Knowledge-Based Systems and Data Science - 29th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2016, Morioka, Japan, August 2-4, 2016, Proceedings*, volume 9799 of *Lecture Notes in Computer Science*, pages 344–355. Springer, 2016.

[14] V. W. Anelli, Y. Deldjoo, T. D. Noia, E. D. Sciascio, and F. A. Merra. Sasha: Semantic-aware shilling attacks on recommender systems exploiting knowledge graphs. In A. Harth, S. Kirrane, A. N. Ngomo, H. Paulheim, A. Rula, A. L. Gentile, P. Haase, and M. Cochez, editors, *The Semantic Web - 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31-June 4, 2020, Proceedings*, volume 12123 of *Lecture Notes in Computer Science*, pages 307–323. Springer, 2020.

[15] V. W. Anelli, T. Di Noia, E. Di Sciascio, A. Ragone, and J. Trotta. Semantic interpretation of top-n recommendations. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[16] V. W. Anelli and T. D. Noia. 2nd workshop on knowledge-aware and conversational recommender systems - kars. In W. Zhu, D. Tao, X. Cheng, P. Cui, E. A. Rundensteiner, D. Carmel, Q. He, and J. X. Yu, editors, *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019.*, pages 3001–3002. ACM, 2019.

[17] V. W. Anelli, T. D. Noia, P. Lops, C. Musto, M. Zanker, P. Basile, D. G. Bridge, and F. Narducci, editors. *Proceedings of the Workshop on Knowledge-aware and Conversational Recommender Systems 2018 co-located with 12th ACM Conference on Recommender Systems, KaRS@RecSys 2018, Vancouver, Canada, October 7, 2018*, volume 2290 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[18] V. W. Anelli, T. D. Noia, P. Lops, and E. D. Sciascio. Feature factorization for top-n recommendation: From item rating to features relevance. In Y. Zheng, W. Pan, S. S. Sahebi, and I. Fernández, editors, *Proceedings of the 1st Workshop on Intelligent Recommender Systems by Knowledge Transfer & Learning co-located with ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, August 27, 2017.*, volume 1887 of *CEUR Workshop Proceedings*, pages 16–21. CEUR-WS.org, 2017.

[19] V. W. Anelli, T. D. Noia, E. D. Sciascio, C. Pomo, and A. Ragone. On the discriminative power of hyper-parameters in cross-validation and how to choose them. In T. Bogers, A. Said, P. Brusilovsky, and D. Tikk, editors, *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2017, Copenhagen, Denmark, September 16-20, 2019.*, pages 447–451. ACM, 2019.

[20] V. W. Anelli, T. D. Noia, E. D. Sciascio, A. Ragone, and J. Trotta. Time-aware personalized popularity in top-n recommendation. In *Workshop on Recommendation in Complex Scenarios co-located with 12th ACM Conference on Recommender Systems (RecSys 2018), Vancouver, BC, Canada, October 2-7, 2018*, 2018.

[21] V. W. Anelli, T. D. Noia, E. D. Sciascio, A. Ragone, and J. Trotta. How to make latent factors interpretable by feeding factorization machines with knowledge graphs. In C. Ghidini, O. Hartig, M. Maleshkova, V. Svátek, I. F. Cruz, A. Hogan, J. Song, M. Lefrançois, and F. Gandon, editors, *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I*, volume 11778 of *Lecture Notes in Computer Science*, pages 38–56. Springer, 2019.

[22] V. W. Anelli, T. D. Noia, E. D. Sciascio, A. Ragone, and J. Trotta. The importance of being dissimilar in recommendation. In C. Hung and G. A. Papadopoulos, editors, *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, SAC 2019, Limassol, Cyprus, April 8-12, 2019*, pages 816–821. ACM, 2019.

[23] V. W. Anelli, T. D. Noia, E. D. Sciascio, A. Ragone, and J. Trotta. Local popularity and time in top-n recommendation. In L. Azzopardi, B. Stein, N. Fuhr, P. Mayr, C. Hauff, and D. Hiemstra, editors, *Advances in Information Retrieval - 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14-18, 2019, Proceedings, Part I*, volume 11437 of *Lecture Notes in Computer Science*, pages 861–868. Springer, 2019.

[24] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[25] K. Bauman, B. Liu, and A. Tuzhilin. Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 717–725. ACM, 2017.

[26] I. Bayer. fastfm: A library for factorization machines. *J. Mach. Learn. Res.*, 17:184:1–184:5, 2016.

[27] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *SIGKDD Explorations*, 9(2):75–79, 2007.

[28] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*, pages 43–52. IEEE Computer Society, 2007.

[29] R. M. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In P. Berkhin, R. Caruana, and X. Wu, editors, *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 95–104. ACM, 2007.

[30] A. Bellogín, I. Cantador, and P. Castells. A study of heterogeneity in recommendations for a social music service. In *Proceedings of the 1st Interna-*

tional Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec '10, pages 1–8, New York, NY, USA, 2010. ACM.

[31] A. Bellogín, P. Castells, and I. Cantador. Precision-oriented evaluation of recommender systems: an algorithmic comparison. In B. Mobasher, R. D. Burke, D. Jannach, and G. Adomavicius, editors, *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*, pages 333–336. ACM, 2011.

[32] A. Bellogín, P. Castells, and I. Cantador. Statistical biases in information retrieval metrics for recommender systems. *Inf. Retr. Journal*, 20(6):606–634, 2017.

[33] A. Bellogín, I. Fernández-Tobías, I. Cantador, and P. Tomeo. Neighbor selection for cold users in collaborative filtering with positive-only feedback. In F. Herrera, S. Damas, R. Montes, S. Alonso, O. Cordón, A. González, and A. Troncoso, editors, *Advances in Artificial Intelligence - 18th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2018, Granada, Spain, October 23-26, 2018, Proceedings*, volume 11160 of *Lecture Notes in Computer Science*, pages 3–12. Springer, 2018.

[34] J. Bennett, S. Lanning, and N. Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[35] S. Berkovsky, T. Kuflik, and F. Ricci. Distributed collaborative filtering with domain specialization. In J. A. Konstan, J. Riedl, and B. Smyth, editors, *Proceedings of the 2007 ACM Conference on Recommender Systems, RecSys 2007, Minneapolis, MN, USA, October 19-20, 2007*, pages 33–40. ACM, 2007.

[36] S. Berkovsky, T. Kuflik, and F. Ricci. Mediation of user models for enhanced personalization in recommender systems. *User Model. User-Adapt. Interact.*, 18(3):245–286, 2008.

[37] T. Berners-Lee, R. T. Fielding, and L. Masinter. Uniform resource identifiers (URI): generic syntax. *RFC*, 2396:1–40, 1998.

[38] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[39] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.

[40] Y. Blanco-Fernández, J. J. Pazos-Arias, A. Gil-Solla, M. R. Cabrer, and M. L. Nores. Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems. *IEEE Trans. Consumer Electronics*, 54(2):727–735, 2008.

[41] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In D. Georgakopoulos and A. Buchmann, editors, *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*, pages 421–430. IEEE Computer Society, 2001.

[42] S. Bostandjiev, J. O'Donovan, and T. Höllerer. Tasteweights: a visual interactive hybrid recommender system. In P. Cunningham, N. J. Hurley, I. Guy, and S. S. Anand, editors, *Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*, pages 35–42. ACM, 2012.

[43] M. R. Bouadjenek, H. Hacid, and M. Bouzeghoub. Social networks and information retrieval, how are they converging? A survey, a taxonomy and an analysis of social information retrieval approaches and platforms. *Inf. Syst.*, 56:1–18, 2016.

[44] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res.*, 21:135–191, 2004.

[45] R. I. Brafman and C. Domshlak. Introducing variable importance tradeoffs into cp-nets. In A. Darwiche and N. Friedman, editors, *UAI '02, Proceedings*

of the 18th Conference in Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta, Canada, August 1-4, 2002, pages 69–76. Morgan Kaufmann, 2002.

[46] J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In G. F. Cooper and S. Moral, editors, *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, University of Wisconsin Business School, Madison, Wisconsin, USA, July 24-26, 1998*, pages 43–52. Morgan Kaufmann, 1998.

[47] G. Brewka, S. Benferhat, and D. L. Berre. Qualitative choice logic. *Artif. Intell.*, 157(1-2):203–237, 2004.

[48] G. Brewka, J. P. Delgrande, J. Romero, and T. Schaub. asprin: Customizing answer set preferences without a headache. In B. Bonet and S. Koenig, editors, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 1467–1474. AAAI Press, 2015.

[49] C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In W. B. Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Dublin, Ireland, 3-6 July 1994 (Special Issue of the SIGIR Forum)*, pages 292–300. ACM/Springer, 1994.

[50] R. D. Burke. Hybrid recommender systems: Survey and experiments. *User Model. User-Adapt. Interact.*, 12(4):331–370, 2002.

[51] R. D. Burke. Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 377–408. Springer, 2007.

[52] P. G. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Model. User-Adapt. Interact.*, 24(1-2):67–119, 2014.

[53] I. Cantador, P. Brusilovsky, and T. Kuflik. Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011). In B. Mobasher, R. D. Burke, D. Jannach, and G. Adomavicius, editors, *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*, pages 387–388. ACM, 2011.

[54] I. Cantador, I. Fernández-Tobías, and A. Bellogín. Relating personality types with user preferences in multiple entertainment domains. In S. Berkovsky, E. Herder, P. Lops, and O. C. Santos, editors, *Late-Breaking Results, Project Papers and Workshop Proceedings of the 21st Conference on User Modeling, Adaptation, and Personalization., Rome, Italy, June 10-14, 2013*, volume 997 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[55] I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi. Cross-domain recommender systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 919–959. Springer, 2015.

[56] B. Cao, N. N. Liu, and Q. Yang. Transfer learning for collective link prediction in multiple heterogenous domains. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 159–166. Omnipress, 2010.

[57] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In Z. Ghahramani, editor, *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 129–136. ACM, 2007.

[58] C. Cappiello, T. D. Noia, B. A. Marcu, and M. Matera. A quality model for linked data exploration. In A. Bozzon, P. Cudré-Mauroux, and C. Pau-

tasso, editors, *Web Engineering - 16th International Conference, ICWE 2016, Lugano, Switzerland, June 6-9, 2016. Proceedings*, volume 9671 of *Lecture Notes in Computer Science*, pages 397–404. Springer, 2016.

[59] P. Castells, N. J. Hurley, and S. Vargas. Novelty and diversity in recommender systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 881–918. Springer, 2015.

[60] S. Chakraborty, R. Tomsett, R. Raghavendra, D. Harborne, M. Alzantot, F. Cerutti, M. B. Srivastava, A. D. Preece, S. Julier, R. M. Rao, T. D. Kelley, D. Braines, M. Sensoy, C. J. Willis, and P. Gurram. Interpretability of deep learning models: A survey of results. In *2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI 2017, San Francisco, CA, USA, August 4-8, 2017*, pages 1–6. IEEE, 2017.

[61] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In D. W. Cheung, I. Song, W. W. Chu, X. Hu, and J. J. Lin, editors, *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 621–630. ACM, 2009.

[62] X. Chen, Z. Qin, Y. Zhang, and T. Xu. Learning to rank features for recommendation over multiple categories. In R. Perego, F. Sebastiani, J. A. Aslam, I. Ruthven, and J. Zobel, editors, *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*, pages 305–314. ACM, 2016.

[63] J. Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.

[64] J. Chomicki. Logical foundations of preference queries. *IEEE Data Eng. Bull.*, 34(2):3–10, 2011.

[65] R. Chung, D. Sundaram, and A. Srinivasan. Integrated personal recommender systems. In M. L. Gini, R. J. Kauffman, D. Sarppo, C. Dellarocas, and F. Dignum, editors, *Proceedings of the 9th International Conference on Electronic Commerce: The Wireless World of Electronic Commerce, 2007, University of Minnesota, Minneapolis, MN, USA, August 19-22, 2007*, volume 258 of *ACM International Conference Proceeding Series*, pages 65–74. ACM, 2007.

[66] C. Cornelio, A. Loreggia, and V. A. Saraswat. Logical conditional preference theories. *CoRR*, abs/1504.06374, 2015.

[67] P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In S. Sen, W. Geyer, J. Freyne, and P. Castells, editors, *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 191–198. ACM, 2016.

[68] H. S. M. Cramer, V. Evers, S. Ramlal, M. van Someren, L. Rutledge, N. Stash, L. Aroyo, and B. J. Wielinga. The effects of transparency on trust in and acceptance of a content-based art recommender. *User Model. User-Adapt. Interact.*, 18(5):455–496, 2008.

[69] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In X. Amatriain, M. Torrens, P. Resnick, and M. Zanker, editors, *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 39–46. ACM, 2010.

[70] P. Cremonesi and M. Quadrana. Cross-domain recommendations without overlapping data: myth or reality? In A. Kobsa, M. X. Zhou, M. Ester, and Y. Koren, editors, *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 297–300. ACM, 2014.

[71] P. Cremonesi, A. Tripodi, and R. Turrin. Cross-domain recommender systems. In M. Spiliopoulou, H. Wang, D. J. Cook, J. Pei, W. Wang, O. R.

Zaïane, and X. Wu, editors, *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on, Vancouver, BC, Canada, December 11, 2011*, pages 496–503. IEEE Computer Society, 2011.

[72] M. F. Dacrema, A. Gasparin, and P. Cremonesi. Deriving item features relevance from collaborative domain knowledge. In V. W. Anelli, T. D. Noia, P. Lops, C. Musto, M. Zanker, P. Basile, D. G. Bridge, and F. Narducci, editors, *Proceedings of the Workshop on Knowledge-aware and Conversational Recommender Systems 2018 co-located with 12th ACM Conference on Recommender Systems, KaRS@RecSys 2018, Vancouver, Canada, October 7, 2018.*, volume 2290 of *CEUR Workshop Proceedings*, pages 1–4. CEUR-WS.org, 2018.

[73] M. de Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro. Semantics-aware content-based recommender systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 119–159. Springer, 2015.

[74] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[75] Y. Deldjoo, V. W. Anelli, H. Zamani, A. Bellogín, and T. Di Noia. A flexible framework for evaluating user and item fairness in recommender systems. *User Modeling and User-Adapted Interaction*, Jan 2021.

[76] A. Dhurandhar, S. Oh, and M. Petrik. Building an interpretable recommender via loss-preserving transformation. *CoRR*, abs/1606.05819, 2016.

[77] Y. Dimopoulos, L. Michael, and F. Athienitou. Ceteris paribus preference elicitation with predictive guarantees. In C. Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 1890–1895, 2009.

[78] C. Domshlak, E. Hüllermeier, S. Kaci, and H. Prade. Preferences in AI: an overview. *Artif. Intell.*, 175(7-8):1037–1052, 2011.

[79] C. Domshlak, E. Hüllermeier, S. Kaci, H. Prade, F. Yaman, T. J. Walsh, M. L. Littman, and M. desJardins. Representing, processing, and learning preferences: Theoretical and practical challenges democratic approximation of lexicographic preference models. *Artif. Intell.*, 175(7):1290–1307, 2011. doi:10.1016/j.artint.2010.11.012.

[80] C. Domshlak and T. Joachims. Unstructuring user preferences: Efficient non-parametric utility revelation. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 169–177. AUAI Press, 2005.

[81] C. Domshlak and T. Joachims. Efficient and non-parametric reasoning over user preferences. *User Model. User-Adapt. Interact.*, 17(1-2):41–69, 2007.

[82] C. Domshlak, S. D. Prestwich, F. Rossi, K. B. Venable, and T. Walsh. Hard and soft constraints for reasoning about qualitative conditional preferences. *J. Heuristics*, 12(4-5):263–285, 2006.

[83] C. Domshlak, F. Rossi, K. B. Venable, and T. Walsh. Reasoning about soft constraints and conditional preferences: complexity results and approximation techniques. In G. Gottlob and T. Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 215–220. Morgan Kaufmann, 2003.

[84] C. N. dos Santos, L. Barbosa, D. Bogdanova, and B. Zadrozny. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 694–699. The Association for Computer Linguistics, 2015.

[85] J. Doyle. Prospects for preferences. *Computational Intelligence*, 20(2):111–136, 2004.

[86] N. Drawel, H. Qu, J. Bentahar, and E. Shakshuki. Specification and automatic verification of trust-based multi-agent systems. *Future Generation Computer Systems*, 2018.

[87] M. J. Dürst and M. Suignard. Internationalized resource identifiers (iris). *RFC*, 3987:1–46, 2005.

[88] A. Elbadrawy and G. Karypis. User-specific feature-based similarity models for top-*n* recommendation of new items. *ACM TIST*, 6(3):33:1–33:20, 2015.

[89] M. Enrich, M. Braunhofer, and F. Ricci. Cold-start management with cross-domain collaborative filtering and tags. In C. Huemer and P. Lops, editors, *E-Commerce and Web Technologies - 14th International Conference, EC-Web 2013, Prague, Czech Republic, August 27-28, 2013. Proceedings*, volume 152 of *Lecture Notes in Business Information Processing*, pages 101–112. Springer, 2013.

[90] R. Fagin. Combining fuzzy information from multiple systems. *J. Comput. Syst. Sci.*, 58(1):83–99, 1999.

[91] R. Falcone, A. Sapienza, and C. Castelfranchi. The relevance of categories for trusting information sources. *ACM Trans. Internet Techn.*, 15(4):13:1–13:21, 2015.

[92] I. Fernández-Tobías and I. Cantador. Exploiting social tags in matrix factorization models for cross-domain collaborative filtering. In T. Bogers, M. Koolen, and I. Cantador, editors, *Proceedings of the 1st Workshop on New Trends in Content-based Recommender Systems co-located with the 8th ACM Conference on Recommender Systems, CBRecSys@RecSys 2014, Foster City, Silicon Valley, California, USA, October 6, 2014.*, volume 1245 of *CEUR Workshop Proceedings*, pages 34–41. CEUR-WS.org, 2014.

[93] I. Fernández-Tobías, I. Cantador, M. Kaminskas, and F. Ricci. A generic semantic-based framework for cross-domain recommendation. In *Proceedings of the 2Nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '11, pages 25–32, New York, NY, USA, 2011. ACM.

[94] I. Fernández-Tobías, I. Cantador, and L. Plaza. An emotion dimensional model based on social tags: Crossing folksonomies and enhancing recommendations. In C. Huemer and P. Lops, editors, *E-Commerce and Web Technologies - 14th International Conference, EC-Web 2013, Prague, Czech Republic, August 27-28, 2013. Proceedings*, volume 152 of *Lecture Notes in Business Information Processing*, pages 88–100. Springer, 2013.

[95] I. Fernández-Tobías, I. Cantador, P. Tomeo, V. W. Anelli, and T. D. Noia. Addressing the user cold start with cross-domain collaborative filtering: exploiting item metadata in matrix factorization. *User Model. User-Adapt. Interact.*, 29(2):443–486, 2019.

[96] I. Fernández-Tobías, P. Tomeo, I. Cantador, T. D. Noia, and E. D. Sciascio. Accuracy and diversity in cross-domain recommendations for cold-start users with positive-only feedback. In S. Sen, W. Geyer, J. Freyne, and P. Castells, editors, *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 119–122. ACM, 2016.

[97] V. Fionda and G. Pirrò. Querying graphs with preferences. In Q. He, A. Iyengar, W. Nejdl, J. Pei, and R. Rastogi, editors, *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 929–938. ACM, 2013.

[98] P. C. Fishburn. *Utility theory for decision making*. Publications in operations research. J. Wiley, 1970.

[99] D. M. Fleder and K. Hosanagar. Recommender systems and their impact on sales diversity. In J. K. MacKie-Mason, D. C. Parkes, and P. Resnick, editors,

Proceedings 8th ACM Conference on Electronic Commerce (EC-2007), San Diego, California, USA, June 11-15, 2007, pages 192–199. ACM, 2007.

[100] S. Flesca and S. Greco. Partially ordered regular languages for graph queries. *J. Comput. Syst. Sci.*, 70(1):1–25, 2005.

[101] S. Funk. Netflix update: Try this at home, 2006.

[102] J. Fürnkranz and E. Hüllermeier, editors. *Preference Learning*. Springer, 2010.

[103] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In M. M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 1606–1611, 2007.

[104] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: a free recommender system library. In B. Mobasher, R. D. Burke, D. Jannach, and G. Adomavicius, editors, *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*, pages 305–308. ACM, 2011.

[105] S. Gao, H. Luo, D. Chen, S. Li, P. Gallinari, and J. Guo. Cross-domain recommendation via cluster-level latent factor model. In H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezný, editors, *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part II*, volume 8189 of *Lecture Notes in Computer Science*, pages 161–176. Springer, 2013.

[106] S. Givon and V. Lavrenko. Predicting social-tags for cold start book recommendations. In L. D. Bergman, A. Tuzhilin, R. D. Burke, A. Felfernig, and L. Schmidt-Thieme, editors, *Proceedings of the 2009 ACM Conference on Recommender Systems, RecSys 2009, New York, NY, USA, October 23-25, 2009*, pages 333–336. ACM, 2009.

[107] O. Goga, H. Lei, S. H. K. Parthasarathi, G. Friedland, R. Sommer, and R. Teixeira. Exploiting innocuous activity for correlating users across sites. In D. Schwabe, V. A. F. Almeida, H. Glaser, R. A. Baeza-Yates, and S. B. Moon, editors, *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 447–458. International World Wide Web Conferences Steering Committee / ACM, 2013.

[108] J. Goldsmith, J. Lang, M. Truszczynski, and N. Wilson. The computational complexity of dominance and consistency in cp-nets. *J. Artif. Intell. Res.*, 33:403–432, 2008.

[109] G. Grahne, A. Thomo, and W. W. Wadge. Preferential regular path queries. *Fundam. Inform.*, 89(2-3):259–288, 2008.

[110] J. T. Guerin, T. E. Allen, and J. Goldsmith. Learning cp-net preferences online from user queries. In P. Perny, M. Pirlot, and A. Tsoukiàs, editors, *Algorithmic Decision Theory - Third International Conference, ADT 2013, Bruxelles, Belgium, November 12-14, 2013, Proceedings*, volume 8176 of *Lecture Notes in Computer Science*, pages 208–220. Springer, 2013.

[111] M. Gueroussova, A. Polleres, and S. A. McIlraith. SPARQL with qualitative and quantitative preferences. In I. Celino, E. D. Valle, M. Krötzsch, and S. Schlobach, editors, *Proceedings of the 2nd International Workshop on Ordering and Reasoning, OrdRing 2013, Co-located with the 12th International Semantic Web Conference (ISWC 2013), Sydney, Australia, October 22nd, 2013*, volume 1059 of *CEUR Workshop Proceedings*, pages 2–8. CEUR-WS.org, 2013.

[112] A. Gunawardana and G. Shani. Evaluating recommender systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 265–308. Springer, 2015.

[113] G. Guo, J. Zhang, Z. Sun, and N. Yorke-Smith. Librec: A java library for recommender systems. In A. I. Cristea, J. Masthoff, A. Said, and N. Tintarev,

editors, *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd Conference on User Modeling, Adaptation, and Personalization (UMAP 2015), Dublin, Ireland, June 29 - July 3, 2015.*, volume 1388 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[114] E. Han and G. Karypis. Feature-based recommendation system. In O. Herzog, H. Schek, N. Fuhr, A. Chowdhury, and W. Teiken, editors, *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, pages 446–452. ACM, 2005.

[115] S. O. Hansson. What is ceteris paribus preference? *J. Philosophical Logic*, 25(3):307–332, 1996.

[116] T. H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Trans. Knowl. Data Eng.*, 15(4):784–796, 2003.

[117] L. He, N. N. Liu, and Q. Yang. Active dual collaborative filtering with both item and attribute feedback. In W. Burgard and D. Roth, editors, *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*. AAAI Press, 2011.

[118] M. He, Q. Meng, and S. Zhang. Collaborative additional variational autoencoder for top-n recommender systems. *IEEE Access*, 7:5707–5713, 2019.

[119] M. He, J. Zhang, P. Yang, and K. Yao. Robust transfer learning for cross-domain collaborative filtering using multiple rating patterns approximation. In Y. Chang, C. Zhai, Y. Liu, and Y. Maarek, editors, *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM 2018, Marina Del Rey, CA, USA, February 5-9, 2018*, pages 225–233. ACM, 2018.

[120] X. He and T. Chua. Neural factorization machines for sparse predictive analytics. In N. Kando, T. Sakai, H. Joho, H. Li, A. P. de Vries, and R. W. White,

editors, *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 355–364. ACM, 2017.

[121] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In F. C. Gey, M. A. Hearst, and R. M. Tong, editors, *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*, pages 230–237. ACM, 1999.

[122] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In W. A. Kellogg and S. Whittaker, editors, *CSCW 2000, Proceeding on the ACM 2000 Conference on Computer Supported Cooperative Work, Philadelphia, PA, USA, December 2-6, 2000*, pages 241–250. ACM, 2000.

[123] J. L. Herlocker, J. A. Konstan, and J. Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Inf. Retr.*, 5(4):287–310, 2002.

[124] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[125] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In S. Sen, W. Geyer, J. Freyne, and P. Castells, editors, *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 241–248. ACM, 2016.

[126] B. Hidasi and D. Tikk. Initializing matrix factorization methods on implicit feedback databases. *J. UCS*, 19(12):1834–1853, 2013.

[127] W. C. Hill, L. Stead, M. Rosenstein, and G. W. Furnas. Recommending and evaluating choices in a virtual community of use. In I. R. Katz, R. L. Mack, L. Marks, M. B. Rosson, and J. Nielsen, editors, *Human Factors in Computing Systems, CHI '95 Conference Proceedings, Denver, Colorado, USA, May 7-11, 1995.*, pages 194–201. ACM/Addison-Wesley, 1995.

[128] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu. Personalized recommendation via cross-domain triadic factorization. In D. Schwabe, V. A. F. Almeida, H. Glaser, R. A. Baeza-Yates, and S. B. Moon, editors, *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 595–606. International World Wide Web Conferences Steering Committee / ACM, 2013.

[129] L. Hu, S. Jian, L. Cao, and Q. Chen. Interpretable recommendation via attraction modeling: Learning multilevel attractiveness over multimodal movie contents. In J. Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 3400–3406. ijcai.org, 2018.

[130] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 263–272. IEEE Computer Society, 2008.

[131] I. Hulpus, N. Prangnawarat, and C. Hayes. Path-based semantic relatedness on linked data and its use to word and entity disambiguation. In M. Arenas, Ó. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. T. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, and S. Staab, editors, *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, volume 9366 of *Lecture Notes in Computer Science*, pages 442–457. Springer, 2015.

[132] P. Jain, P. Kumaraguru, and A. Joshi. @i seek 'fb.me': identifying users across multiple online social networks. In L. Carr, A. H. F. Laender, B. F.

Lóscio, I. King, M. Fontoura, D. Vrandecic, L. Aroyo, J. P. M. de Oliveira, F. Lima, and E. Wilde, editors, *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*, pages 1259–1268. International World Wide Web Conferences Steering Committee / ACM, 2013.

[133] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems - An Introduction*. Cambridge University Press, 2010.

[134] G. Jawaheer, M. Szomszor, and P. Kostkova. Comparison of implicit and explicit feedback from an online music recommendation service. In *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '10, pages 47–51, New York, NY, USA, 2010. ACM.

[135] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments - part 1. *Inf. Process. Manage.*, 36(6):779–808, 2000.

[136] K. S. Jones, S. Walker, and S. E. Robertson. A probabilistic model of information retrieval: development and comparative experiments - part 2. *Inf. Process. Manage.*, 36(6):809–840, 2000.

[137] Y. Juan, Y. Zhuang, W. Chin, and C. Lin. Field-aware factorization machines for CTR prediction. In S. Sen, W. Geyer, J. Freyne, and P. Castells, editors, *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 43–50. ACM, 2016.

[138] S. Kabbur, X. Ning, and G. Karypis. FISM: factored item similarity models for top-n recommender systems. In I. S. Dhillon, Y. Koren, R. Ghani, T. E. Senator, P. Bradley, R. Parekh, J. He, R. L. Grossman, and R. Uthurusamy, editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 659–667. ACM, 2013.

[139] S. Kalloori and F. Ricci. Improving cold start recommendation by mapping feature-based preferences to item comparisons. In M. Bieliková, E. Herder, F. Cena, and M. C. Desmarais, editors, *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, UMAP 2017, Bratislava, Slovakia, July 09 - 12, 2017*, pages 289–293. ACM, 2017.

[140] M. Kaminskas, I. Fernández-Tobías, I. Cantador, and F. Ricci. Ontology-based identification of music for places. In L. Cantoni and Z. P. Xiang, editors, *Information and Communication Technologies in Tourism 2013, ENTER 3013, Proceedings of the International Conference in Innsbruck, Austria, January 22-25, 2013.*, pages 436–447. Springer, 2013.

[141] G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management, Atlanta, Georgia, USA, November 5-10, 2001*, pages 247–254. ACM, 2001.

[142] H. Khrouf and R. Troncy. Hybrid event recommendation using linked data and user diversity. In Q. Yang, I. King, Q. Li, P. Pu, and G. Karypis, editors, *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 185–192. ACM, 2013.

[143] W. Kießling. Foundations of preferences in database systems. In *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002, Hong Kong, August 20-23, 2002*, pages 311–322. Morgan Kaufmann, 2002.

[144] W. Kießling, M. Endres, and F. Wenzel. The preference SQL system - an overview. *IEEE Data Eng. Bull.*, 34(2):11–18, 2011.

[145] D. Kim and B. Yum. Collaborative filtering based on iterative principal component analysis. *Expert Syst. Appl.*, 28(4):823–830, 2005.

[146] D. Kluver and J. A. Konstan. Evaluating recommender behavior for new users. In A. Kobsa, M. X. Zhou, M. Ester, and Y. Koren, editors, *Eighth*

ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014, pages 121–128. ACM, 2014.

[147] J. A. Konstan, S. M. McNee, C. Ziegler, R. Torres, N. Kapoor, and J. Riedl. Lessons on applying automated recommender systems to information-seeking tasks. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 1630–1633. AAAI Press, 2006.

[148] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Y. Li, B. Liu, and S. Sarawagi, editors, *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 426–434. ACM, 2008.

[149] Y. Koren and R. M. Bell. Advances in collaborative filtering. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 77–118. Springer, 2015.

[150] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[151] Y. Koren and J. Sill. Ordrec: an ordinal model for predicting personalized item rating distributions. In B. Mobasher, R. D. Burke, D. Jannach, and G. Adomavicius, editors, *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*, pages 117–124. ACM, 2011.

[152] F. Koriche and B. Zanuttini. Learning conditional preference networks. *Artif. Intell.*, 174(11):685–703, 2010.

[153] A. B. Kouki. *Performance prediction and evaluation in recommender systems: An information retrieval perspective*. PhD thesis, Universidad Autónoma de Madrid, 2012.

[154] J. Lees-Miller, F. Anderson, B. Hoehn, and R. Greiner. Does wikipedia information help netflix predictions? In M. A. Wani, X. Chen, D. Casasent, L. A. Kurgan, T. Hu, and K. Hafeez, editors, *Seventh International Conference on Machine Learning and Applications, ICMLA 2008, San Diego, California, USA, 11-13 December 2008*, pages 337–343. IEEE Computer Society, 2008.

[155] A. Leonardo and F. Vasco. OWLPref: Uma representação declarativa de preferências para web semântica. Anais do XXVII Congresso da SBC, pages 1411–1420, 2007.

[156] B. Li, Q. Yang, and X. Xue. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In C. Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 2052–2057, 2009.

[157] B. Li, Q. Yang, and X. Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In A. P. Danyluk, L. Bottou, and M. L. Littman, editors, *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 617–624. ACM, 2009.

[158] C. Li, M. A. Soliman, K. C. Chang, and I. F. Ilyas. Ranksql: Supporting ranking queries in relational database management systems. In K. Böhm, C. S. Jensen, L. M. Haas, M. L. Kersten, P. Larson, and B. C. Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 1342–1345. ACM, 2005.

[159] X. Li, H. Fang, Q. Yang, and J. Zhang. Who is your best friend?: Ranking social network friends according to trust relationship. In T. Mitrovic, J. Zhang, L. Chen, and D. Chin, editors, *Proceedings of the 26th Conference*

on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018, pages 301–309. ACM, 2018.

[160] X. Li, E. Gavves, C. G. M. Snoek, M. Worring, and A. W. M. Smeulders. Personalizing automated image annotation using cross-entropy. In K. S. Candan, S. Panchanathan, B. Prabhakaran, H. Sundaram, W. Feng, and N. Sebe, editors, *Proceedings of the 19th International Conference on Multimedia 2011, Scottsdale, AZ, USA, November 28 - December 1, 2011*, pages 233–242. ACM, 2011.

[161] X. Li and J. She. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 305–314. ACM, 2017.

[162] X. Li, C. G. M. Snoek, and M. Worring. Unsupervised multi-feature tag relevance learning for social image retrieval. In S. Li, X. Gao, and N. Sebe, editors, *Proceedings of the 9th ACM International Conference on Image and Video Retrieval, CIVR 2010, Xi'an, China, July 5-7, 2010*, pages 10–17. ACM, 2010.

[163] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[164] H. Liu, X. He, F. Feng, L. Nie, R. Liu, and H. Zhang. Discrete factorization machines for fast feature-based recommendation. In J. Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 3449–3455. ijcai.org, 2018.

[165] J. Liu, Z. Li, J. Tang, Y. Jiang, and H. Lu. Personalized geo-specific tag recommendation for photos on social websites. *IEEE Trans. Multimedia*, 16(3):588–600, 2014.

[166] J. Liu, J. Shi, W. Cai, B. Liu, W. Pan, Q. Yang, and Z. Ming. Transfer learning from APP domain to news domain for dual cold-start recommendation. In Y. Zheng, W. Pan, S. S. Sahebi, and I. Fernández, editors, *Proceedings of the 1st Workshop on Intelligent Recommender Systems by Knowledge Transfer & Learning co-located with ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, August 27, 2017.*, volume 1887 of *CEUR Workshop Proceedings*, pages 38–41. CEUR-WS.org, 2017.

[167] J. Liu, Y. Xiong, C. Wu, Z. Yao, and W. Liu. Learning conditional preference networks from inconsistent examples. *IEEE Trans. Knowl. Data Eng.*, 26(2):376–390, 2014.

[168] J. Liu, Z. Yao, Y. Xiong, W. Liu, and C. Wu. Learning conditional preference network from noisy samples using hypothesis testing. *Knowl.-Based Syst.*, 40:7–16, 2013.

[169] N. N. Liu, M. Zhao, and Q. Yang. Probabilistic latent preference analysis for collaborative filtering. In D. W. Cheung, I. Song, W. W. Chu, X. Hu, and J. J. Lin, editors, *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*, pages 759–766. ACM, 2009.

[170] W. Liu, C. Wu, B. Feng, and J. Liu. Conditional preference in recommender systems. *Expert Syst. Appl.*, 42(2):774–788, 2015.

[171] B. Loni, Y. Shi, M. Larson, and A. Hanjalic. Cross-domain collaborative filtering with factorization machines. In M. de Rijke, T. Kenter, A. P. de Vries, C. Zhai, F. de Jong, K. Radinsky, and K. Hofmann, editors, *Advances in Information Retrieval - 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings*, volume 8416 of *Lecture Notes in Computer Science*, pages 656–661. Springer, 2014.

[172] S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In G. M. Olson and

R. Jeffries, editors, *Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006*, pages 1097–1101. ACM, 2006.

[173] R. Meymandpour and J. G. Davis. Enhancing recommender systems using linked open data-based semantic analysis of items. In J. G. Davis and A. Bozzon, editors, *3rd Australasian Web Conference, AWC 2015, Sydney, Australia, January 2015*, volume 166 of *CRPIT*, pages 11–17. Australian Computer Society, 2015.

[174] S. E. Middleton, N. Shadbolt, and D. D. Roure. Ontological user profiling in recommender systems. *ACM Trans. Inf. Syst.*, 22(1):54–88, 2004.

[175] D. Milne and I. H. Witten. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceedings of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy*, pages 25–30. AAAI Press, 2008.

[176] N. Mirbakhsh and C. X. Ling. Improving top-n recommendation for cold-start users via cross-domain information. *TKDD*, 9(4):33:1–33:19, 2015.

[177] M. Montaner, B. López, and J. L. de la Rosa. A taxonomy of recommender agents on the internet. *Artif. Intell. Rev.*, 19(4):285–330, 2003.

[178] C. Musto, P. Basile, P. Lops, M. de Gemmis, and G. Semeraro. Linked open data-enabled strategies for top-n recommendations. In T. Bogers, M. Koolen, and I. Cantador, editors, *Proceedings of the 1st Workshop on New Trends in Content-based Recommender Systems co-located with the 8th ACM Conference on Recommender Systems, CBRecSys@RecSys 2014, Foster City, Silicon Valley, California, USA, October 6, 2014.*, volume 1245 of *CEUR Workshop Proceedings*, pages 49–56. CEUR-WS.org, 2014.

[179] C. Musto, P. Basile, P. Lops, M. de Gemmis, and G. Semeraro. Introducing linked open data in graph-based recommender systems. *Inf. Process. Manage.*, 53(2):405–435, 2017.

[180] C. Musto, P. Lops, P. Basile, M. de Gemmis, and G. Semeraro. Semantics-aware graph-based recommender systems exploiting linked open data. In J. Vassileva, J. Blustein, L. Aroyo, and S. K. D'Mello, editors, *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, UMAP 2016, Halifax, NS, Canada, July 13 - 17, 2016*, pages 229–237. ACM, 2016.

[181] C. Musto, P. Lops, M. de Gemmis, and G. Semeraro. Semantics-aware recommender systems exploiting linked open data and graph-based features. In P. Champin, F. L. Gandon, M. Lalmas, and P. G. Ipeirotis, editors, *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, pages 457–460. ACM, 2018.

[182] C. Musto, F. Narducci, P. Lops, M. de Gemmis, and G. Semeraro. Explod: A framework for explaining recommendations based on the linked open data cloud. In S. Sen, W. Geyer, J. Freyne, and P. Castells, editors, *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 151–154. ACM, 2016.

[183] C. Musto, G. Semeraro, P. Lops, M. de Gemmis, and F. Narducci. Leveraging social media sources to generate personalized music playlists. In C. Huemer and P. Lops, editors, *E-Commerce and Web Technologies - 13th International Conference, EC-Web 2012, Vienna, Austria, September 4-5, 2012. Proceedings*, volume 123 of *Lecture Notes in Business Information Processing*, pages 112–123. Springer, 2012.

[184] M. Naseri, M. Elahi, and P. Cremonesi. Investigating the decision making process of users based on the polimovie dataset. In M. Ge and F. Ricci, editors, *Proceedings of the 2nd International Workshop on Decision Making and Recommender Systems, Bolzano, Italy, October 22-23, 2015.*, volume 1533 of *CEUR Workshop Proceedings*, pages 41–44. CEUR-WS.org, 2015.

[185] M. Nasery, M. Braunhofer, and F. Ricci. Recommendations with optimal combination of feature-based and item-based preferences. In J. Vassileva,

J. Blustein, L. Aroyo, and S. K. D'Mello, editors, *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, UMAP 2016, Halifax, NS, Canada, July 13 - 17, 2016*, pages 269–273. ACM, 2016.

[186] X. Ning and G. Karypis. Sparse linear methods with side information for top-n recommendations. In P. Cunningham, N. J. Hurley, I. Guy, and S. S. Anand, editors, *Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*, pages 155–162. ACM, 2012.

[187] T. D. Noia. LOSM: a SPARQL endpoint to query open street map. In S. Villata, J. Z. Pan, and M. Dragoni, editors, *Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015.*, volume 1486 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[188] T. D. Noia, T. Lukasiewicz, M. V. Martínez, G. I. Simari, and O. Tifrea-Marciuska. Combining existential rules with the power of cp-theories. In Q. Yang and M. J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2918–2925. AAAI Press, 2015.

[189] T. D. Noia, C. Magarelli, A. Maurino, M. Palmonari, and A. Rula. Using ontology-based data summarization to develop semantics-aware recommender systems. In A. Gangemi, R. Navigli, M. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, and M. Alam, editors, *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*, volume 10843 of *Lecture Notes in Computer Science*, pages 128–144. Springer, 2018.

[190] T. D. Noia, R. Mirizzi, V. C. Ostuni, and D. Romito. Exploiting the web of data in model-based recommender systems. In P. Cunningham, N. J. Hurley, I. Guy, and S. S. Anand, editors, *Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*, pages 253–256. ACM, 2012.

[191] T. D. Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker. Linked open data to support content-based recommender systems. In V. Presutti and H. S. Pinto, editors, *I-SEMANTICS 2012 - 8th International Conference on Semantic Systems, I-SEMANTICS '12, Graz, Austria, September 5-7, 2012*, pages 1–8. ACM, 2012.

[192] T. D. Noia and V. C. Ostuni. Recommender systems and linked open data. In W. Faber and A. Paschke, editors, *Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Berlin, Germany, July 31 - August 4, 2015, Tutorial Lectures*, volume 9203 of *Lecture Notes in Computer Science*, pages 88–113. Springer, 2015.

[193] T. D. Noia, V. C. Ostuni, J. Rosati, P. Tomeo, E. D. Sciascio, R. Mirizzi, and C. Bartolini. Building a relatedness graph from linked open data: A case study in the IT domain. *Expert Syst. Appl.*, 44:354–366, 2016.

[194] T. D. Noia, V. C. Ostuni, P. Tomeo, and E. D. Sciascio. Sprank: Semantic path-based ranking for top-*N* recommendations using linked open data. *ACM TIST*, 8(1):9:1–9:34, 2016.

[195] I. Nunes, S. Miles, M. Luck, and C. J. P. de Lucena. An introduction to reasoning over qualitative multi-attribute preferences. *Knowledge Eng. Review*, 30(3):342–372, 2015.

[196] S. Oramas, V. C. Ostuni, T. D. Noia, X. Serra, and E. D. Sciascio. Sound and music recommendation with knowledge graphs. *ACM TIST*, 8(2):21:1–21:21, 2017.

[197] F. Ortega, J. L. Sánchez, J. Bobadilla, and A. Gutiérrez. Improving collaborative filtering-based recommender systems results using pareto dominance. *Inf. Sci.*, 239:50–61, 2013.

[198] V. C. Ostuni, T. D. Noia, E. D. Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In Q. Yang, I. King,

Q. Li, P. Pu, and G. Karypis, editors, *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 85–92. ACM, 2013.

[199] E. Palumbo, G. Rizzo, and R. Troncy. entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In P. Cremonesi, F. Ricci, S. Berkovsky, and A. Tuzhilin, editors, *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, pages 32–36. ACM, 2017.

[200] W. Pan. A survey of transfer learning for collaborative recommendation with auxiliary data. *Neurocomputing*, 177:447–453, 2016.

[201] W. Pan, N. N. Liu, E. W. Xiang, and Q. Yang. Transfer learning to predict missing ratings via heterogeneous user feedbacks. In T. Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2318–2323. IJCAI/AAAI, 2011.

[202] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang. Transfer learning in collaborative filtering for sparsity reduction. In M. Fox and D. Poole, editors, *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press, 2010.

[203] E. Pantaleo, V. W. Anelli, T. D. Noia, and G. Sérasset. Etytree: A graphical and interactive etymology dictionary based on wiktionary. In R. Barrett, R. Cummings, E. Agichtein, and E. Gabrilovich, editors, *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017*, pages 1635–1640. ACM, 2017.

[204] U. Paquet and N. Koenigstein. One-class collaborative filtering with random graphs. In D. Schwabe, V. A. F. Almeida, H. Glaser, R. A. Baeza-Yates, and S. B. Moon, editors, *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 999–1008. International World Wide Web Conferences Steering Committee / ACM, 2013.

[205] D. Parra and X. Amatriain. Walk the talk - analyzing the relation between implicit and explicit feedback for preference elicitation. In J. A. Konstan, R. Conejo, J. Marzo, and N. Oliver, editors, *User Modeling, Adaption and Personalization - 19th International Conference, UMAP 2011, Girona, Spain, July 11-15, 2011. Proceedings*, volume 6787 of *Lecture Notes in Computer Science*, pages 255–268. Springer, 2011.

[206] A. Passant. dbrec - music recommendations using dbpedia. In P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Z. Pan, I. Horrocks, and B. Glimm, editors, *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part II*, volume 6497 of *Lecture Notes in Computer Science*, pages 209–224. Springer, 2010.

[207] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[208] H. Paulheim and J. Fürnkranz. Unsupervised generation of data mining features from linked open data. In D. D. Burdescu, R. Akerkar, and C. Badica, editors, *2nd International Conference on Web Intelligence, Mining and Semantics, WIMS '12, Craiova, Romania, June 6-8, 2012*, pages 31:1–31:12. ACM, 2012.

[209] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2007.

[210] G. Peake and J. Wang. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In Y. Guo and F. Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2060–2069. ACM, 2018.

[211] G. Piao and J. G. Breslin. Measuring semantic distance for linked open data-enabled recommender systems. In S. Ossowski, editor, *Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016*, pages 315–320. ACM, 2016.

[212] G. Pigozzi, A. Tsoukiàs, and P. Viappiani. Preferences in artificial intelligence. *Ann. Math. Artif. Intell.*, 77(3-4):361–401, 2016.

[213] I. Pilászy, D. Zibriczky, and D. Tikk. Fast als-based matrix factorization for explicit and implicit feedback datasets. In X. Amatriain, M. Torrens, P. Resnick, and M. Zanker, editors, *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 71–78. ACM, 2010.

[214] Z. Qian, P. Zhong, and R. Wang. Tag refinement for user-contributed images via graph learning and nonnegative tensor factorization. *IEEE Signal Process. Lett.*, 22(9):1302–1305, 2015.

[215] A. Rana and D. Bridge. Explanation chains: Recommendations by explanation. In D. Tikk and P. Pu, editors, *Proceedings of the Poster Track of the 11th ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, August 28, 2017.*, volume 1905 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2017.

[216] S. Rendle. Factorization machines. In G. I. Webb, B. Liu, C. Zhang, D. Gunopulos, and X. Wu, editors, *ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14-17 December 2010*, pages 995–1000. IEEE Computer Society, 2010.

[217] S. Rendle. *Context-Aware Ranking with Factorization Models*, volume 330 of *Studies in Computational Intelligence*. Springer, 2011.

[218] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In J. A. Bilmes and A. Y. Ng, editors, *UAI 2009, Proceedings of the Twenty-Fifth Conference on*

Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009, pages 452–461. AUAI Press, 2009.

[219] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In M. Rappa, P. Jones, J. Freire, and S. Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 811–820. ACM, 2010.

[220] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In W. Ma, J. Nie, R. A. Baeza-Yates, T. Chua, and W. B. Croft, editors, *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 635–644. ACM, 2011.

[221] S. Rendle and L. Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In B. D. Davison, T. Suel, N. Craswell, and B. Liu, editors, *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, pages 81–90. ACM, 2010.

[222] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[223] F. Ricci, L. Rokach, and B. Shapira, editors. *Recommender Systems Handbook*. Springer, 2015.

[224] P. Ristoski, J. Rosati, T. D. Noia, R. D. Leone, and H. Paulheim. Rdf2vec: RDF graph embeddings and their applications. *Semantic Web*, 10(4):721–752, 2019.

[225] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[226] J. Rosati, T. D. Noia, T. Lukasiewicz, R. D. Leone, and A. Maurino. Preference queries with ceteris paribus semantics for linked data. In C. Debruyne, H. Panetto, R. Meersman, T. S. Dillon, G. Weichhart, Y. An, and C. A. Ardagna, editors, *On the Move to Meaningful Internet Systems: OTM 2015 Conferences - Confederated International Conferences: CoopIS, ODBASE, and C&TC 2015, Rhodes, Greece, October 26-30, 2015, Proceedings*, volume 9415 of *Lecture Notes in Computer Science*, pages 423–442. Springer, 2015.

[227] M. Rowe. Semanticsvd++: Incorporating semantic taste evolution for predicting ratings. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Warsaw, Poland, August 11-14, 2014 - Volume II*, pages 213–220. IEEE Computer Society, 2014.

[228] N. Sachdeva, K. Gupta, and V. Pudi. Attentive neural architecture incorporating song features for music recommendation. In S. Pera, M. D. Ekstrand, X. Amatriain, and J. O'Donovan, editors, *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 417–421. ACM, 2018.

[229] S. Sahebi and P. Brusilovsky. Cross-domain collaborative recommendation in a cold-start context: The impact of user profile size on the quality of recommendation. In S. Carberry, S. Weibelzahl, A. Micarelli, and G. Semeraro, editors, *User Modeling, Adaptation, and Personalization - 21th International Conference, UMAP 2013, Rome, Italy, June 10-14, 2013, Proceedings*, volume 7899 of *Lecture Notes in Computer Science*, pages 289–295. Springer, 2013.

[230] S. Sahebi, P. Brusilovsky, and V. Bobrokov. Cross-domain recommendation for large-scale data. In Y. Zheng, W. Pan, S. S. Sahebi, and I. Fernández, editors, *Proceedings of the 1st Workshop on Intelligent Recommender Systems by Knowledge Transfer & Learning co-located with ACM Conference on Recommender Systems (RecSys 2017), Como, Italy, August 27, 2017.*, volume 1887 of *CEUR Workshop Proceedings*, pages 9–15. CEUR-WS.org, 2017.

[231] A. Said and A. Bellogín. Comparative recommender system evaluation: benchmarking recommendation frameworks. In A. Kobsa, M. X. Zhou, M. Ester, and Y. Koren, editors, *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 129–136. ACM, 2014.

[232] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1257–1264. Curran Associates, Inc., 2007.

[233] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[234] J. Sang, C. Xu, and J. Liu. User-aware image tag refinement via ternary semantic analysis. *IEEE Trans. Multimedia*, 14(3-2):883–895, 2012.

[235] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system - a case study. In *ACM WebKDD 2000 Workshop*. ACM SIGKDD, 2000.

[236] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *EC*, pages 158–167, 2000.

[237] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In V. Y. Shen, N. Saito, M. R. Lyu, and M. E. Zurko, editors, *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 285–295. ACM, 2001.

[238] N. Sawant, R. Datta, J. Li, and J. Z. Wang. Quest for relevant tags using local interaction networks and visual content. In J. Z. Wang, N. Boujemaa, N. O. Ramirez, and A. Natsev, editors, *Proceedings of the 11th ACM SIGMM International Conference on Multimedia Information Retrieval, MIR 2010, Philadelphia, Pennsylvania, USA, March 29-31, 2010*, pages 231–240. ACM, 2010.

[239] J. B. Schafer, D. Frankowski, J. L. Herlocker, and S. Sen. Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 291–324. Springer, 2007.

[240] M. Schmachtenberg, T. Strufe, and H. Paulheim. Enhancing a location-based recommendation system by enrichment with structured data from the web. In R. Akerkar, N. Bassiliades, J. Davies, and V. Ermolayev, editors, *4th International Conference on Web Intelligence, Mining and Semantics (WIMS 14), WIMS '14, Thessaloniki, Greece, June 2-4, 2014*, pages 17:1–17:12. ACM, 2014.

[241] N. Shadbolt, T. Berners-Lee, and W. Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, 2006.

[242] B. Shapira, L. Rokach, and S. Freilikhman. Facebook single and cross domain data for recommendation systems. *User Model. User-Adapt. Interact.*, 23(2-3):211–247, 2013.

[243] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In I. R. Katz, R. L. Mack, L. Marks, M. B.

Rosson, and J. Nielsen, editors, *Human Factors in Computing Systems, CHI '95 Conference Proceedings, Denver, Colorado, USA, May 7-11, 1995.*, pages 210–217. ACM/Addison-Wesley, 1995.

[244] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, and A. Hanjalic. xclimf: optimizing expected reciprocal rank for data with multiple levels of relevance. In Q. Yang, I. King, Q. Li, P. Pu, and G. Karypis, editors, *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 431–434. ACM, 2013.

[245] Y. Shi, M. Larson, and A. Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In X. Amatriain, M. Torrens, P. Resnick, and M. Zanker, editors, *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 269–272. ACM, 2010.

[246] Y. Shi, M. Larson, and A. Hanjalic. Tags as bridges between domains: Improving recommendation with tag-induced cross-domain collaborative filtering. In J. A. Konstan, R. Conejo, J. Marzo, and N. Oliver, editors, *User Modeling, Adaption and Personalization - 19th International Conference, UMAP 2011, Girona, Spain, July 11-15, 2011. Proceedings*, volume 6787 of *Lecture Notes in Computer Science*, pages 305–316. Springer, 2011.

[247] A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3145–3153. PMLR, 2017.

[248] W. Siberski, J. Z. Pan, and U. Thaden. Querying the semantic web with preferences. In I. F. Cruz, S. Decker, D. Allemang, C. Preist, D. Schwabe, P. Mika, M. Uschold, and L. Aroyo, editors, *The Semantic Web - ISWC 2006, 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA,*

November 5-9, 2006, Proceedings, volume 4273 of *Lecture Notes in Computer Science*, pages 612–624. Springer, 2006.

[249] R. R. Sinha and K. Swearingen. The role of transparency in recommender systems. In L. G. Terveen and D. R. Wixon, editors, *Extended abstracts of the 2002 Conference on Human Factors in Computing Systems, CHI 2002, Minneapolis, Minnesota, USA, April 20-25, 2002*, pages 830–831. ACM, 2002.

[250] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In M. J. Silva, A. H. F. Laender, R. A. Baeza-Yates, D. L. McGuinness, B. Olstad, Ø. H. Olsen, and A. O. Falcão, editors, *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, pages 623–632. ACM, 2007.

[251] E. Smyth. *Would the Internet widen public participation?* PhD thesis, University of Leeds, United Kingdom, 2012.

[252] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 2960–2968, 2012.

[253] H. Steck. Evaluation of recommendations: rating-prediction and ranking. In Q. Yang, I. King, Q. Li, P. Pu, and G. Karypis, editors, *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 213–220. ACM, 2013.

[254] A. Stewart, E. Diaz-Aviles, W. Nejdl, L. B. Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Cross-tagging for personalized open social networking.

In C. Cattuto, G. Ruffo, and F. Menczer, editors, *HYPERTEXT 2009, Proceedings of the 20th ACM Conference on Hypertext and Hypermedia, Torino, Italy, June 29 - July 1, 2009*, pages 271–278. ACM, 2009.

[255] Z. Sun, J. Yang, J. Zhang, A. Bozzon, L. Huang, and C. Xu. Recurrent knowledge graph embedding for effective recommendation. In S. Pera, M. D. Ekstrand, X. Amatriain, and J. O'Donovan, editors, *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, pages 297–305. ACM, 2018.

[256] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos. Justified recommendations based on content and rating data. In *WebKDD Workshop on Web Mining and Web Usage Analysis*, 2008.

[257] M. Szomszor, H. Alani, I. Cantador, K. O'Hara, and N. Shadbolt. Semantic modelling of user interests based on cross-folksonomy analysis. In A. P. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. W. Finin, and K. Thirunarayan, editors, *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*, volume 5318 of *Lecture Notes in Computer Science*, pages 632–648. Springer, 2008.

[258] M. Szomszor, I. Cantador, and H. Alani. Correlating user profiles from multiple folksonomies. In P. Brusilovsky and H. C. Davis, editors, *HYPERTEXT 2008, Proceedings of the 19th ACM Conference on Hypertext and Hypermedia, Pittsburgh, PA, USA, June 19-21, 2008*, pages 33–42. ACM, 2008.

[259] G. Takács and D. Tikk. Alternating least squares for personalized ranking. In P. Cunningham, N. J. Hurley, I. Guy, and S. S. Anand, editors, *Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*, pages 83–90. ACM, 2012.

[260] A. Taneja and A. Arora. Cross domain recommendation using multidimensional tensor factorization. *Expert Syst. Appl.*, 92:304–316, 2018.

[261] N. Tintarev and J. Masthoff. A survey of explanations in recommender systems. In *Proceedings of the 23rd International Conference on Data Engineering Workshops, ICDE 2007, 15-20 April 2007, Istanbul, Turkey*, pages 801–810. IEEE Computer Society, 2007.

[262] N. Tintarev and J. Masthoff. Designing and evaluating explanations for recommender systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 479–510. Springer, 2011.

[263] A. Tiroshi, S. Berkovsky, M. A. Kâafar, T. Chen, and T. Kuflik. Cross social networks interests predictions based ongraph features. In Q. Yang, I. King, Q. Li, P. Pu, and G. Karypis, editors, *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 319–322. ACM, 2013.

[264] A. Tiroshi and T. Kuflik. Domain ranking for cross domain collaborative filtering. In J. Masthoff, B. Mobasher, M. C. Desmarais, and R. Nkambou, editors, *User Modeling, Adaptation, and Personalization - 20th International Conference, UMAP 2012, Montreal, Canada, July 16-20, 2012. Proceedings*, volume 7379 of *Lecture Notes in Computer Science*, pages 328–333. Springer, 2012.

[265] A. Troumpoukis, S. Konstantopoulos, and A. Charalambidis. An extension of SPARQL for expressing qualitative preferences. In C. d'Amato, M. Fernández, V. A. M. Tamma, F. Lécué, P. Cudré-Mauroux, J. F. Sequeda, C. Lange, and J. Heflin, editors, *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, volume 10587 of *Lecture Notes in Computer Science*, pages 711–727. Springer, 2017.

[266] K. H. L. Tso and L. Schmidt-Thieme. Evaluation of attribute-aware recommender system algorithms on data with varying characteristics. In W. K. Ng, M. Kitsuregawa, J. Li, and K. Chang, editors, *Advances in Knowledge Discovery and Data Mining, 10th Pacific-Asia Conference, PAKDD 2006,*

Singapore, April 9-12, 2006, Proceedings, volume 3918 of *Lecture Notes in Computer Science*, pages 831–840. Springer, 2006.

[267] T. X. Tuan and T. M. Phuong. 3d convolutional networks for session-based recommendation with content features. In P. Cremonesi, F. Ricci, S. Berkovsky, and A. Tuzhilin, editors, *Proceedings of the Eleventh ACM Conference on Recommender Systems, RecSys 2017, Como, Italy, August 27-31, 2017*, pages 138–146. ACM, 2017.

[268] S. Vargas, L. Baltrunas, A. Karatzoglou, and P. Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In A. Kobsa, M. X. Zhou, M. Ester, and Y. Koren, editors, *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 209–216. ACM, 2014.

[269] S. Vargas and P. Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In B. Mobasher, R. D. Burke, D. Jannach, and G. Adomavicius, editors, *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*, pages 109–116. ACM, 2011.

[270] J. Vig, S. Sen, and J. Riedl. Tagsplanations: explaining recommendations using tags. In C. Conati, M. Bauer, N. Oliver, and D. S. Weld, editors, *Proceedings of the 14th International Conference on Intelligent User Interfaces, IUI 2009, Sanibel Island, Florida, USA, February 8-11, 2009*, pages 47–56. ACM, 2009.

[271] M. Vlachos, C. Dünner, R. Heckel, V. G. Vassiliadis, T. P. Parnell, and K. Atasu. Addressing interpretability and cold-start in matrix factorization for recommender systems. *IEEE Trans. Knowl. Data Eng.*, 31(7):1253–1266, 2019.

[272] G. H. Von Wright. *The Logic of Preference*. Edinburgh University Press, 1963.

[273] V. Walter Anelli, R. De Leone, T. Di Noia, T. Lukasiewicz, and J. Rosati. Combining rdf and sparql with cp-theories to reason about preferences in a linked data setting. *Semantic Web*, pages 1–29, 12 2018.

[274] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo. Multi-task feature learning for knowledge graph enhanced recommendation. In L. Liu, R. W. White, A. Mantrach, F. Silvestri, J. J. McAuley, R. S. Baeza-Yates, and L. Zia, editors, *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2000–2010. ACM, 2019.

[275] T. Wang, Y. Cai, H. Leung, Z. Cai, and H. Min. Entropy-based term weighting schemes for text categorization in VSM. In *27th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2015, Vietri sul Mare, Italy, November 9-11, 2015*, pages 325–332. IEEE Computer Society, 2015.

[276] X. Wang, X. He, F. Feng, L. Nie, and T. Chua. TEM: tree-enhanced embedding model for explainable recommendation. In P. Champin, F. L. Gandon, M. Lalmas, and P. G. Ipeirotis, editors, *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1543–1552. ACM, 2018.

[277] M. Weimer, A. Karatzoglou, and A. J. Smola. Improving maximum margin matrix factorization. *Machine Learning*, 72(3):263–276, 2008.

[278] S. Weng and M. Liu. Feature-based recommendations for one-to-one marketing. *Expert Syst. Appl.*, 26(4):493–508, 2004.

[279] N. Wilson. Extending cp-nets with stronger conditional preference statements. In D. L. McGuinness and G. Ferguson, editors, *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, pages 735–741. AAAI Press / The MIT Press, 2004.

[280] N. Wilson. Computational techniques for a simple theory of conditional preferences. *Artif. Intell.*, 175(7-8):1053–1091, 2011.

[281] P. Winoto and T. Y. Tang. If you like the devil wears prada the book, will you also enjoy the devil wears prada the movie? A study of cross-domain recommendations. *New Generation Comput.*, 26(3):209–225, 2008.

[282] W. S. Wong, R. W. P. Luk, H. V. Leong, L. K. Ho, and D. L. Lee. Re-examining the effects of adding relevance information in a relevance feedback environment. *Inf. Process. Manage.*, 44(3):1086–1116, 2008.

[283] C. Wongchokprasitti, J. Peltonen, T. Ruotsalo, P. Bandyopadhyay, G. Jacucci, and P. Brusilovsky. User model in a box: Cross-system user model transfer for resolving cold start problems. In F. Ricci, K. Bontcheva, O. Conlan, and S. Lawless, editors, *User Modeling, Adaptation and Personalization - 23rd International Conference, UMAP 2015, Dublin, Ireland, June 29 - July 3, 2015. Proceedings*, volume 9146 of *Lecture Notes in Computer Science*, pages 289–301. Springer, 2015.

[284] Q. Wu, C. J. C. Burges, K. M. Svore, and J. Gao. Adapting boosting for information retrieval measures. *Inf. Retr.*, 13(3):254–270, 2010.

[285] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T. Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In C. Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 3119–3125. ijcai.org, 2017.

[286] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In W. Kraaij, A. P. de Vries, C. L. A. Clarke, N. Fuhr, and N. Kando, editors, *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 391–398. ACM, 2007.

[287] J. Yang, Z. Sun, A. Bozzon, and J. Zhang. Learning hierarchical feature influence for recommendation by recursive regularization. In S. Sen, W. Geyer, J. Freyne, and P. Castells, editors, *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, pages 51–58. ACM, 2016.

[288] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: a heterogeneous information network approach. In B. Carterette, F. Diaz, C. Castillo, and D. Metzler, editors, *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*, pages 283–292. ACM, 2014.

[289] F. Yuan, G. Guo, J. M. Jose, L. Chen, H. Yu, and W. Zhang. Boostfm: Boosted factorization machines for top-n feature-based recommendation. In G. A. Papadopoulos, T. Kuflik, F. Chen, C. Duarte, and W. Fu, editors, *Proceedings of the 22nd International Conference on Intelligent User Interfaces, IUI 2017, Limassol, Cyprus, March 13-16, 2017*, pages 45–54. ACM, 2017.

[290] M. Zanker. The influence of knowledgeable explanations on users' perception of a recommender system. In P. Cunningham, N. J. Hurley, I. Guy, and S. S. Anand, editors, *Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*, pages 269–272. ACM, 2012.

[291] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2016.

[292] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W. Ma. Collaborative knowledge base embedding for recommender systems. In B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discov-*

ery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, pages 353–362. ACM, 2016.

[293] Q. Zhang, D. Wu, J. Lu, F. Liu, and G. Zhang. A cross-domain recommender system with consistent information transfer. *Decision Support Systems*, 104:49–63, 2017.

[294] Y. Zhang. Explainable recommendation: Theory and applications. *CoRR*, abs/1708.06409, 2017.

[295] Y. Zhang, B. Cao, and D. Yeung. Multi-domain collaborative filtering. In P. Grünwald and P. Spirtes, editors, *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, July 8-11, 2010*, pages 725–732. AUAI Press, 2010.

[296] Y. Zhang and X. Chen. Explainable recommendation: A survey and new perspectives. *CoRR*, abs/1804.11192, 2018.

[297] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *The 37th Int. Conf. on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia*, pages 83–92, 2014.

[298] Y. Zhang, M. Zhang, Y. Zhang, G. Lai, Y. Liu, H. Zhang, and S. Ma. Daily-aware personalized recommendation based on feature-level time series analysis. In A. Gangemi, S. Leonardi, and A. Panconesi, editors, *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 1373–1383. ACM, 2015.

[299] L. Zhao, S. J. Pan, and Q. Yang. A unified framework of active transfer learning for cross-system recommendation. *Artif. Intell.*, 245:38–55, 2017.

[300] W. X. Zhao, S. Li, Y. He, L. Wang, J. Wen, and X. Li. Exploring demographic information in social media for product recommendation. *Knowl. Inf. Syst.*, 49(1):61–89, 2016.

[301] Y. Zheng, B. Mobasher, and R. D. Burke. Incorporating context correlation into context-aware matrix factorization. In D. Jannach, J. Mengin, B. Mobasher, A. Passerini, and P. Viappiani, editors, *Proceedings of the IJCAI 2015 Joint Workshop on Constraints and Preferences for Configuration and Recommendation and Intelligent Techniques for Web Personalization co-located with the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015), Buenos Aires, Argentina, July 27, 2015.*, volume 1440 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2015.

[302] F. Zhu, Y. Wang, C. Chen, G. Liu, M. A. Orgun, and J. Wu. A deep framework for cross-domain and cross-system recommendations. In J. Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.*, pages 3711–3717. ijcai.org, 2018.

[303] F. Zhuang, P. Luo, H. Xiong, Y. Xiong, Q. He, and Z. Shi. Cross-domain learning from multiple sources: A consensus regularization perspective. *IEEE Trans. Knowl. Data Eng.*, 22(12):1664–1678, 2010.

[304] C. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In A. Ellis and T. Hagino, editors, *Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005*, pages 22–32. ACM, 2005.