PGCOMP - Programa de Pós-Graduação em Ciência da Computação
Universidade Federal da Bahia (UFBA)
Av. Milton Santos, s/n - Ondina
Salvador, BA, Brasil, 40170-110

https://pgcomp.ufba.br
pgcomp@ufba.br

Linked Open Data (LOD) is a cloud of freely accessible and interconnected datasets encompassing machine-readable data. These data are available under open Semantic Web standards, such as RDF and SPARQL. One notable example of a LOD set is DBpedia, a crowd-sourced community effort to extract structured information from Wikipedia and make this information openly available on the Web. The semantic content of LOD and the advanced features of SPARQL has opened unprecedented opportunities for enabling semantic-aware applications. LOD-based Recommender Systems (RSs) usually leverage the data available within LOD datasets such as DBpedia to recommend items such as movies, places, books, and music to end-users. These systems use a semantic similarity algorithm that calculates the degree of matching between pairs of resources in the RDF graph, by counting the number of direct and indirect links between them, the length of the path between them, or the hierarchy of classes. Conversely, calculating similarity in RDF graphs could be difficult because each resource can have hundreds of links to other nodes. Not all of them are semantically relevant or can be applied to all resources in the graph. This can lead to the well-known matrix sparsity problem. Nevertheless, some effort has been made to select subsets of features, i.e., links, which are more helpful to computing similarity between items of a graph dataset, reducing the matrix dimension. Despite several studies in this field, there is still a lack of solutions applied to the personalization of feature selection tasks. In this context, we propose personalized strategies to improve semantic similarity precision in LOD-based Recommender Systems, including i) applying a feature selection approach to filter the best features for a particular user; ii) personalizing the RDF graph by adding weights to the edges, according to the user's previous preferences; and iii) exploiting the similarity of literal properties as well as the links from the user model. The evaluation experiments used combined data from DBpedia and MovieLens and DBpedia and LastFM datasets. Results indicate significant increases in top-n recommendation tasks in Precision@K (K=5, 10), Map, and NDCG over non-personalized baseline similarities methods such as Linked Data Semantic Distance (LDSD) and Resource Similarity (ReSim). The results show that the strategies proposed in this work can be effective in improving semantic recommendation systems in various knowledge domains, as the solution is scalable to any LOD-based databases.

Keywords: Recommender Systems; Linked Open Data; Semantic Similarity; Personalization; Feature Selection

DSC | 044 | 2023 Exploiting Lod-Based Similarity Personalization Strategies for Recommender Systems Gabriela Oliveira Mota da Silva

UFBA

Exploiting Lod-Based Similarity Personalization Strategies for Recommender Systems

Gabriela Oliveira Mota da Silva

Tese de Doutorado

Universidade Federal da Bahia

Programa de Pós-Graduação em
Ciência da Computação

Setembro | 2023

Universidade Federal da Bahia
Instituto de Computação

Programa de Pós-Graduação em Ciência da Computação

EXPLOITING LOD-BASED SIMILARITY PERSONALIZATION STRATEGIES FOR RECOMMENDER SYSTEMS

Gabriela Oliveira Mota da Silva

TESE DE DOUTORADO

Salvador
28 de setembro de 2023

GABRIELA OLIVEIRA MOTA DA SILVA

EXPLOITING LOD-BASED SIMILARITY PERSONALIZATION STRATEGIES FOR RECOMMENDER SYSTEMS

Esta Tese de Doutorado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Doutor em Ciência da Computação.

Orientador: Frederico Araújo Durão

Salvador
28 de setembro de 2023

Ficha catalográfica elaborada pela Biblioteca Universitária de
Ciências e Tecnologias Prof. Omar Catunda, SIBI – UFBA.

S586 Silva, Gabriela Oliveira Mota da

Exploiting Lod-Based Similarity Personalization Strategies for Recommender Systems/ Gabriela Oliveira Mota da Silva. – Salvador, 2023.

126 f.

Orientador: Prof. Dr. Frederico Araújo Durão

Tese (Doutorado) – Universidade Federal da Bahia. Instituto de Computação, 2023.

1. Recommender Systems. 2. Linked Open Data. 3. Semantic Similarity. 4. Personalization. 5. Feature Selection. I. Durão, Frederico Araújo. II. Universidade Federal da Bahia. III. Título.

CDU 004

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DA BAHIA
INSTITUTO DE COMPUTAÇÃO
PGCOMP - Programa de Pós-Graduação em Ciência da Computação
http://pgcomp.ufba.br

Gabriela Oliveira Mota da Silva

Exploiting Lod-based Similarity Personalization Strategies for Recommender Systems

Esta tese foi julgada adequada à obtenção do título de Doutor em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da UFBA.

Salvador, 28 de setembro de 2023

Frederico Araujo Durão

Prof. Dr. Frederico Araújo Durão (Orientador - UFBA)

gov.br Documento assinado digitalmente
NATASHA CORREIA QUEIROZ LINO
Data: 02/10/2023 09:05:46-0300
Verifique em https://validar.iti.gov.br

Profa. Dra. Natasha Correia Queiroz Lino (UFPB)

[illegible]

Prof. Dr. Rosalvo Ferreira de Oliveira Neto (UNIVASF)

[illegible]

Profa. Dra. Daniela Barreiro Claro (UFBA)

Laís do Nascimento Salvador

Profa. Dra. Laís do Nascimento Salvador (UFBA)

I dedicate this thesis to my mother, who always encouraged my sister and me to study and pursue our dreams. Her love and dedication have now turned into precious achievements.

ACKNOWLEDGEMENTS

I'd like to begin my acknowledgments by apologizing in advance for being so long, but it is necessary to put the circumstances in which this work was completed into context. Many might say that it took too much time, but I say that it took strictly the necessary time. Shortly after my return from my PhD stay in Canada, the pandemic began. During this terrible period, I suffered the irreparable losses of my maternal grandfather, my father, and my paternal grandmother. These sudden losses triggered a series of emotional factors that were sufficient to destabilize me and take my focus off my academic life.

Having said that, I would like to acknowledge my advisor Professor Frederico Araujo Durão. It is no overstatement to say that it is thanks to his effort in not giving up on me that this thesis exists and can add a spark to the scientific knowledge of humanity. Even at times when I was furthest away from my duties as a PhD student, he was understanding enough to give me the time I needed to get back on track emotionally.

He was also responsible for awakening in me a love for teaching and research. All of this has built a relationship of respect and admiration in me, not only for his technical work but also for his character and humanity. Thank you, Fred, for guiding me along this path and making me a better researcher and person. You are such an inspiration to me and to so many others who dream of making a difference in the lives of other students.

I could not fail to thank my partner José Diógenes Pereira Torres, who pulled me out of the depths I had sunk into and gave me the greatest gift of my life, our little baby daughter. Thank you for being my source of daily care, emotional support, and love. Thank you for being my home! And to my daughter Diana Maria, thank you for choosing me as your mother and giving me the opportunity for a new beginning. I love you both more than anything!

I am deeply grateful to all the PGCOMP professors at UFBA's Institute of Computing for their teaching and daily inspiration. I am also very grateful to my colleagues in the RECSYS Research Group, who actively participated in the development of my thesis and throughout my academic life. My special thanks go to Diogo Vinícius, João Paulo Dias, and the German exchange student Marian "João", my constant companions on conference trips and kind support in difficult times. I want to acknowledge the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) for funding the project. Special thanks to Professor Miriam Capretz, Professor of Software Engineering in the Department of Electrical and Computer Engineering at Western University, who looked after me during my PhD stay (2019) in London, Canada.

I dedicate a special acknowledgment to my mother Auristela Oliveira Mota, who strongly believes in the power of education. She never spared any effort to guarantee me the best possible education. She encouraged me to read from an early age and molded me to seek knowledge. She also fought with me, encouraged me and gave me strength.

For that, I will be eternally grateful. As well as my mother, I would like to thank my younger sister, Gisele Oliveira Mota da Silva, for all her unconditional love and support. It has always been the three of us against the world!

“Begin at the beginning”, the King said gravely, “and go on till you come to the end: then stop.”

—LEWIS CARROLL (Alice in Wonderland)

RESUMO

Dados Abertos Conectados (Linked Open Data - LOD, em Inglês) é uma nuvem de bancos de dados interconectados, de livre acesso e legíveis por máquina, pois estão disponíveis em padrões abertos da Web Semântica, como RDF e SPARQL. Um exemplo relevante de banco LOD é a DBpedia, uma iniciativa comunitária para extrair informações estruturadas da Wikipedia e disponibilizá-las abertamente na Web. O conteúdo semântico disponibilizado pelos dados abertos conectados e os recursos avançados da linguagem SPARQL permitiram o desenvolvimento de aplicativos sensíveis à semântica. Os sistemas de recomendação (em Inglês: Recommender Systems - RS) baseados em LOD geralmente aproveitam os dados de bancos LOD, e.g. DBpedia, para recomendar itens como filmes, lugares, livros e músicas aos usuários finais. Esses sistemas usam um algoritmo de similaridade semântica que calcula o grau de correspondência entre pares de recursos do grafo RDF, contando o número de links diretos e indiretos entre eles, o comprimento do caminho entre eles ou analisando a hierarquia de suas classes. Por outro lado, calcular a similaridade em grafos RDF pode ser difícil porque cada recurso pode ter centenas de links para outros nós e nem todos eles são semanticamente relevantes ou podem ser aplicados a todos os recursos do grafo. Isso pode levar ao conhecido problema de esparsidade da matriz. No entanto, é possível selecionar subconjuntos de características que são mais úteis para calcular a semelhança entre itens de um grafo, reduzindo a dimensão da matriz. Apesar de vários estudos nesse campo, ainda faltam soluções aplicadas à personalização da etapa de seleção de características (Feature Selection - FS, em Inglês). Nesse contexto, propomos estratégias personalizadas para melhorar a precisão da similaridade semântica em sistemas de recomendação baseados em LOD, incluindo i) a aplicação de uma abordagem de seleção de características para filtrar as melhores propriedades para um usuário específico; ii) a personalização do grafo RDF adicionando pesos às arestas, de acordo com as preferências anteriores do usuário; e iii) a exploração da similaridade das propriedades literais do modelo do usuário. Os experimentos de avaliação usaram dados combinados dos bancos de dados MovieLens e LastFM com os dados semânticos da DBpedia. Os resultados indicam aumentos estatisticamente significativos nas recomendações top-n em todas as métricas testadas: Precision@K (K=5, 10), Map e NDCG, em relação aos métodos de similaridade de referência não personalizados, como Linked Data Semantic Distance (LDSD) e Resource Similarity (ReSim). Os resultados mostram que as estratégias propostas neste trabalho podem ser eficientes para aprimorar sistemas de recomendação semânticos em diversos domínios do conhecimento, pois a solução é escalável para quaisquer bancos de dados baseados em LOD.

Palavras-chave: Sistemas de Recomendação, Dados Abertos Conectados, Similaridade Semântica, Personalização, Seleção de Características.

ABSTRACT

Linked Open Data (LOD) is a cloud of freely accessible and interconnected datasets encompass machine-readable data. These data are available under open Semantic Web standards, such as Resource Description Framework (RDF) and SPARQL Protocol and RDF Query Language (SPARQL). One notable example of a LOD set is DBpedia, a crowd-sourced community effort to extract structured information from Wikipedia and make this information openly available on the Web. The semantic content of LOD and the advanced features of SPARQL has opened unprecedented opportunities for enabling semantic-aware applications. LOD-based Recommender Systems Recommender Systems usually leverage the data available within LOD datasets such as DBpedia to recommend items such as movies, places, books, and music to end-users. These systems use a semantic similarity algorithm that calculates the degree of matching between pairs of resources in the RDF graph, by counting the number of direct and indirect links between them, the length of the path between them, or the hierarchy of classes. Conversely, calculating similarity in RDF graphs could be difficult because each resource can have hundreds of links to other nodes. Not all of them are semantically relevant or can be applied to all resources in the graph. This can lead to the well-known matrix sparsity problem. Nevertheless, some effort has been made to select subsets of features, i.e., links, which are more helpful to computing similarity between items of a graph dataset, reducing the matrix dimension. Despite several studies in this field, there is still a lack of solutions applied to the personalization of feature selection tasks. In this context, we propose personalized strategies to improve semantic similarity precision in LOD-based Recommender Systems, including i) applying a feature selection approach to filter the best features for a particular user; ii) personalizing the RDF graph by adding weights to the edges, according to the user's previous preferences; and iii) exploiting the similarity of literal properties as well as the links from the user model. The evaluation experiments used combined data from DBpedia and MovieLens and DBpedia and LastFM datasets. Results indicate significant increases in top-n recommendation tasks in Precision@K (K=5, 10), Map, and NDCG over non-personalized baseline similarities methods such as Linked Data Semantic Distance (LDSD) and Resource Similarity (ReSim). The results show that the strategies proposed in this work can be effective in improving semantic recommendation systems in various knowledge domains, as the solution is scalable to any LOD-based databases.

Keywords: Recommender Systems, Linked Open Data, Semantic Similarity, Personalization, Feature Selection.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

API	Application Programming Interface	56
CF	Collaborative Filtering	35
FR	Feature Ranking	44
FS	Feature Selection	56
FOAF	Friend of a Friend	15
HTTP	HyperText Transfer Protocol	21
KG	Knowledge Graph	48
LDSD	Linked Data Semantic Distance	87
LiSim	Literals Similarity	88
LOD	Linked Open Data	49
MAE	Mean Absolute Error	37
MAP	Mean Average Precision	76
MRR	Mean Reciprocal Rank	38
NDCG	Normalized Discounted Cumulative Gain	76
OKG	Open Knowledge Graph	1
OWL	Web Ontology Language	13
PLDSD	Personalized Linked Data Semantic Distance	89
RDF	Resource Description Framework	40
RDFS	Resource Description Framework Schema	14
ReSim	Resource Similarity	87
RS	Recommender System	9
RSs	Recommender Systems	27
RMSE	Root Mean Square Error	36
RR	Reciprocal Rank	38
SPARQL	SPARQL Protocol and RDF Query Language	1
URI	Uniform Resource Identifier	56
VSM	Vector Space Model	
W3C	World Wide Web Consortium	20

Chapter 1

INTRODUCTION

While the World Wide Web has provided means for creating a Web of human-readable documents, the Web of Data (also referred to as Linked Data) has paved the way for a Web of structured and machine-readable data. Linked Open Data (LOD) — which graph is illustrated in Figure 1.1 — is a powerful blend of Linked Data that provides access to a large and increasing amount of diverse data structured under open Semantic Web standards, such as Resource Description Framework (RDF)[1] and SPARQL Protocol and RDF Query Language (SPARQL)[2], the query language for retrieving and manipulating RDF data (BERNERS-LEE, 2009). The LOD cloud currently contains 1,594 datasets with more than 20,000 links (as of May 2023). These datasets are organized by colors in 9 categories, as Figure 1.1 shows, according to the type of knowledge they describe, which are: Cross Domain, Geography, Government, Life Sciences, Linguistics, Media, Publications, Social Networking, and User Generated.

One notable example of a LOD set is DBpedia, a crowd-sourced community effort to extract structured information from Wikipedia[3] and make this information openly available on the Web in a machine-readable form as a giant Open Knowledge Graph (OKG) (LEHMANN et al., 2015). DBpedia is classified as a cross-domain LOD knowledge base, as are all the other light brown-coloured datasets in the Figure 1.1. In 2020, the English version of the DBpedia dataset described 7.62 million entities (HOLZE, 2023).

The semantic content of LOD datasets and the advanced features of SPARQL has opened unprecedented opportunities for the development of semantic aware applications, including Recommender Systems. LOD-based Recommender Systems (RSs) usually leverage the data available within Linked Open Data datasets, such as DBpedia, to recommend items — for example, movies, books, and music — to the end users (PASSANT, 2010; NOIA et al., 2012; PIAO; BRESLIN, 2016). These systems use a semantic similarity algorithm that calculates the degree of matching between pairs of Linked Data resources. Because RDF represents data as a graph, these algorithms, in general, count

[1]https://www.w3.org/RDF/
[2]https://www.w3.org/TR/rdf-sparql-query/
[3]https://www.wikipedia.org/

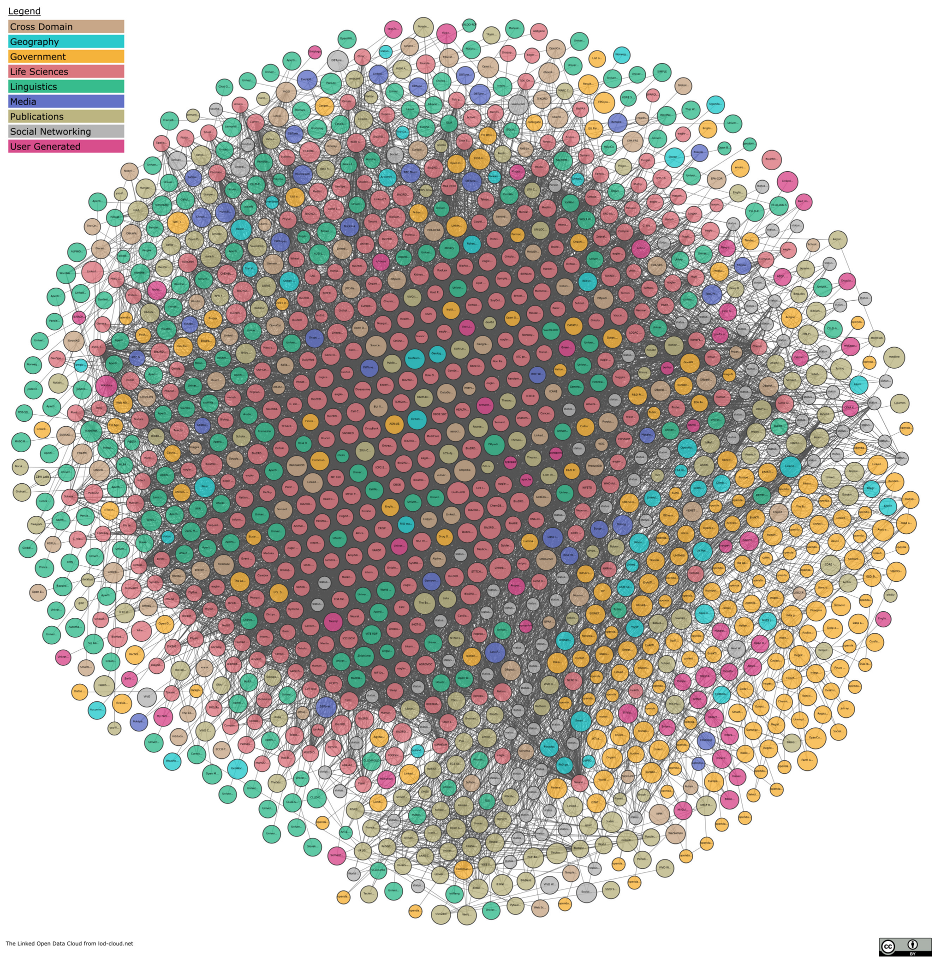


Figure 1.1: The Linked Open Data Cloud (MCCRAE, 2023).

the number of direct and indirect links — i.e., the edges in the graph —, count the length of the path between two resources, or their place in the hierarchy of classes (PASSANT, 2010; PIAO; ARA; BRESLIN, 2016; CHENIKI et al., 2016).

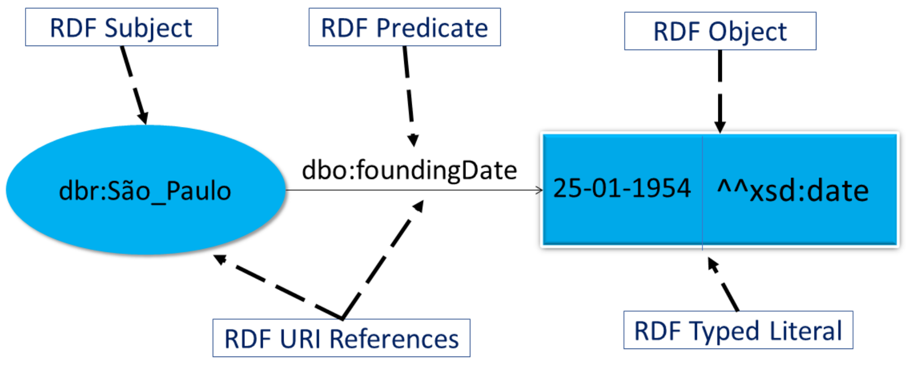


Figure 1.2: RDF structure example.

What semantic-aware applications have in common is that they rely on an RDF graph architecture. The RDF statement (also known as *triples*) appears in the form of *subject-predicate-object* as shown in Figure 1.2. Subjects, predicates, and objects are uniquely identifiable using Uniform Resource Identifiers (URIs). The subject corresponds to the resource to which the statement refers. The predicate indicates a *property* that the resource possesses, which could be one of two types: *object property*, when the property links one resource to another, and *datatype property*, when the property links one resource to a *literal*, such as a string or a number. Objects, therefore, are the resources or literals that correspond to values of properties. Moreover, considering that a subject can be connected to several objects to express various statements and that each object (from each statement) can also be the subject of another statement, the RDF graph representation creates a vast dataset of interconnected nodes. In the Linked Open Data (LOD) project, several of those RDF datasets are interconnected in a huge network of structured knowledge, as seen before in Figure 1.1.

1.1 MOTIVATION

Recommender Systems (RSs) are software tools and techniques that solve the information overload problem by suggesting items most likely interesting to a particular user (RICCI; ROKACH; SHAPIRA, 2015). The growing importance of the Internet as a platform for digital and commercial transactions has driven the development of recommender system technology. One of the main factors behind this trend is the easy way users can express their preferences over the Internet. Take Netflix, for example. Users can conveniently provide feedback with the click of a mouse. The standard feedback mechanism is usually in the form of ratings, where users assign numerical values from a predefined system, such as a five-star system, or binary values, such as in a like/dislike system, to describe their preferences for different items (AGGARWAL, 2016).

The basic principle underlying recommendation algorithms' work is that significant dependencies exist between user- and item-centric activity. For example, a user interested in a historical documentary is likelier to be interested in another historical documentary or an educational program than in an action movie. These algorithms are called content-based recommendation methods. According to Aggarwal (2016), one problem with this approach is that most users would have viewed only a small fraction of the large universe of available movies. As a result, most of the ratings are unspecified. This leads to a known RS problem: the cold-start problem, mainly when a new item or user is added to the system.

Another approach for calculating recommendations is through collaborative filtering methods, in which the basic idea is that these unspecified ratings can be imputed because the observed ratings are often highly correlated across various users and items (AGGARWAL, 2016). For example, consider two users, Alice and Bob, who have similar tastes. If the ratings, which both have specified, are very similar, then the system can infer Alice's missing ratings based on Bob's ratings of the same items. Most of the methods that have been proposed in the literature to deal with the cold start problem are related to collaborative filtering systems (LAM et al., 2008; ZHANG et al., 2010; GUO, 2012).

Usually, when describing an item, its attributes are extracted from metadata associated with it. However, the content extracted from metadata is often too brief to accurately define user interests, while the use of textual features involves a number of complications due to natural language ambiguity. In an attempt to overcome these challenges, recent research has introduced semantic techniques that changes from a keyword-based approach to a concept-based approach for representing items and user profiles (GEMMIS et al., 2015). These are semantic-aware systems, also known as LOD-based recommendation systems, in which the structure of the RDF graph is used to calculate semantic similarity and recommend items to the user, instead of analysing textual metadata.

There are various approaches in the literature for dealing with the task of accounting for semantic similarity, such as developing algorithms that count the number of direct and indirect links — i.e., the edges in the graph — (PASSANT, 2010); count the length of the path between two resources — i.e., the nodes — (PIAO; ARA; BRESLIN, 2016); and count the place of the resources in the hierarchy of classes (CHENIKI et al., 2016), among others. However, calculating similarity in RDF graphs is a complex task, as a single node can have hundreds of links to other nodes, and not all are semantically relevant. Also, those properties are not applied to all the resources of the graph. It leads the system to the well-known sparsity problem of Content-Based Recommender Systems. Some effort has been made minimize this problem by selecting subsets of links that seem more helpful in computing the similarity between items of a graph dataset (CATHERINE; COHEN, 2016; MUSTO et al., 2016), reducing the matrix dimension. Recent studies cite several methods to find and exclude or rank features based on diverse goals — see Section 3.8 in Chapter 3 for examples.

Feature Selection (FS) is a critical task in recommender systems that aims to identify the most relevant features or attributes of the items recommended. Although, none of those mentioned above methods approaches from a user-personalized perspective. Features are commonly filtered by classic methods such as i) Correlation analysis, which involves examining the correlation between each feature and the target variable; ii) Information Gain, which is a measure of how much the feature contributes to predicting the target variable; and iii) Principal component analysis (PCA), that involves transforming the original feature set into a lower-dimensional space that captures the most significant variance in the data (GEMMIS et al., 2015). And although some work involves the automated feature selection (NOIA et al., 2018), most of the processes used to date require manual curation of the database by experts in the domain of study.

To this point, the literature shows that there is still room for research into automatic feature selection methods that combine with the personalization of features according to the specific interests of the target user. Existing work poorly explores the role of the user in the FS phase of a recommendation system, which is usually at the beginning of the system's operating flow. In addition, there is a lack of research that addresses the cold-start and sparsity matrix problems through a content-centered vision of system items, using a personalization strategy based on the user's previous choices.

1.2 PROBLEM STATEMENT

One common characteristic shared by previous works on LOD similarity measures is that they consider all the links in an RDF graph as having similar importance. At the most, they perform a FS method to rank the main features based only on the system's domain in a domain knowledge-driven way. This means domain-specific knowledge about the items can be used to select the most relevant features. For example, artist, genre, and album are important characteristics in the music domain, but irrelevant in a book recommendation system.

This standardized way of classifying features does not take advantage of the unique semantic vision of a particular user. Personalization of features in content-based user models is still poorly exploited in the literature. Taking the movie context as an example, most existing methods select features based on the semantics they represent in the movie domain. Thus, features such as the movie genre, the release date, the director, the persons starring in the movie, among others, are chosen as the subset of relevant features and are considered to be of similar importance to the user.

For example, a user named Bob watches movies of various genres - he has eclectic taste - but he only watches the latest released movies. The release year is often excluded from datasets by FS tasks either because it is not considered a relevant characteristic or because it is represented by a property of type literal instead of a link. Following this reasoning, it means that a movie that is a new release will possibly not be recommended to Bob. This leads to the following questions: Can the recommendation algorithms consider all properties as having the same importance for all users? What difference would it make in the recommendations for Bob if the movie genre was excluded from the set of relevant characteristics and the release year was not?

In addition to the selection of features, diverse similarity measures were proposed to operate recommendations through LOD-based systems (PASSANT, 2010; NOIA et al., 2012; PIAO; BRESLIN, 2016). They can generally handle both the sparsity and the cold-start problem well, when concerning to collaborative-filtering environment. Automated content-based methods that focus on previous item-user interactions are necessary to solve problems in the research area. Currently, none of the systems studied for this research (see Chapter 4) manipulate features in a specific way for each user.

Meymandpour & Davis (2016) and Musto et al. (2016) reviewed existing semantic similarity measures and developed feature-based methods and statistical approaches to improve the performance of graph-based recommendation algorithms. Although they explored various feature selection techniques, their methods could benefit from personalizing the user model to provide more accurate recommendations. Even more recent works follow the same line of thought. Natarajan et al. (2022) calculate the closeness of items between domains by exploiting the semantic relationship rather than the similarity between the attributes of various resources. They proposed a model that provides personalized recommendations for the target new user with the user preferences obtained from the source domain and by exploiting item semantic relatedness. Unlike our work, they do not rank properties according to user preferences, but focus on solving the cold start problem by looking into a general similarity model.

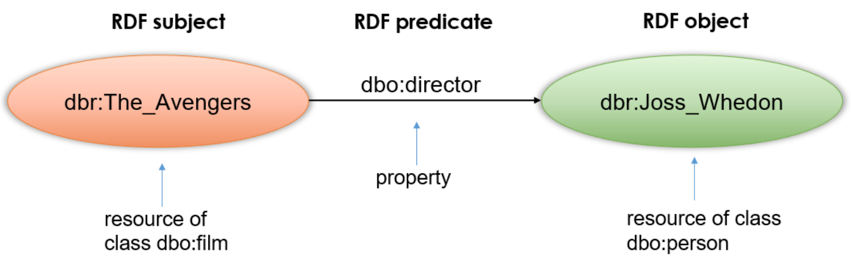


Figure 1.3: RDF example showing a node-to-node relationship.

In LOD-based recommender systems, item features are expressed by links between nodes - which represent the features' *properties* - in the RDF graph. Thus, similarity algorithms exploit the semantics of these links to determine the relations between pairs of nodes in the graph. Many semantic relationships are expressed by a node-to-node connection as seen in Figure 1.3, where the triple ⟨ dbr:The_Avengers, dbo:director, dbr:Joss_Whedon ⟩ express who is the person — resource of class dbo:person — that directed that film. Although, features also can be expressed by literal values, as text or numbers, like in the triple ⟨ dbr:Sao_Paulo, dbo:foundingDate, 1954-01-25(xsd:date) ⟩, shown in Figure 1.2. State-of-the-art work in literature poorly explores the semantics present on the literal values. This ends up excluding potentially important properties for the user from the recommendation calculations, as in another example of a user who hates watching long films. For her, the dbo:runtime property is very important when deciding which film to watch. As current algorithms don't consider literal properties, triples formed by ⟨ dbo:Film, dbo:runtime, xsd:double ⟩ are not computed, causing a semantic loss.

The semantic loss caused by excluding literal properties from recommendation calculations becomes more relevant if we analyse the proportion between object properties and literal properties within DBpedia. According to statistics[4], the DBpedia 2015-10 ontology encompasses 1,596 properties with typed literal values against 1,099 properties with reference values, i.e. object properties. This gives a ratio of approximately 45% more literal properties present in the database. A significant amount of work on semantic recommender systems using DBpedia could benefit from an algorithm that takes into account literal properties. For instance, Durão & Bridge (2018) propose a Linked Data browser powered by an iterative classification algorithm. They model the user's profile with Linked Data to personalize the recommendations within a given neighbourhood of a Linked Data graph. If the system were able to capture the semantics of the literal properties, the recommendations could be more assertive from the user's point of view.

[4]The most recent statistics found are from the 2015 version of DBpedia. Source: <https://downloads.dbpedia.org/wiki-archive/dbpedia-dataset-version-2015-10.html>

1.2.1 Research Questions

The gaps in the literature discussed above guided the development of the following research questions:

- Q1: Can the system be more precise in recommending items to the user if it calculates the similarity using literal and link properties?
- Q2: Can the system recommend items better suited to the user's taste by exploiting her preferred properties?
- Q3: Can we automatically pick the properties that influence the user's choices regardless of the domain?
- Q4: Is the proposed link ranking method feasible to personalize the system?
- Q5: Can the system benefit from a preprocessing step that filters the domain-relevant features before entering the user personalization method?

1.3 GOAL

Inspired by the aforementioned challenges, this thesis aims to **exploit and propose personalized methods for calculating Semantic Similarity in LOD-based Recommender Systems that lead to more accurate recommendations**. This is made through various automatized approaches to personalize the user model, such as i) assigning weights to the links in the LOD graph; ii) selecting the best features; and iii) exploiting the similarity of literal properties as well as of the object properties.

These approaches aim to minimize the sparsity problem since we rank and select the features that will be computed for recommendations. In addition, we aim to solve the cold-start problem when a new item is added to the system, as the feature ranking task is based on the user's previous preferences about the features that both old items and the new item share. This is possible because we use domain-specific semantic knowledge graphs, i.e. all the triples in the main graph are formed by subjects of the same class (rdf:type). For instance, if we are recommending films, then we capture a portion of DBpedia to build a knowledge graph where all triples have subjects of type dbo:Film. We explain better about the structure provided by the Resource Description Framework Schema (RDFS) in Chapter 2.

The RDF graphs are personalized using the approaches described above to validate if the goals were achieved. Then, the personalized user models are combined in a Recommender System with non-personalized baseline similarity methods, such as the Linked Data Semantic Distance (LDSD) (PASSANT, 2010), the Summarization FS (NOIA et al., 2018), and the Resource Similarity (ReSim) (PIAO; ARA; BRESLIN, 2016). In addition, we explain our generic recommender model in which other existing LOD-enabled semantic measures can be combined to achieve the goals of recommender systems in diverse knowledge domains.

The evaluation shows whether using these personalization strategies impacts ranking accuracy in the context of LOD-based Recommender Systems. We use a movie dataset from MovieLens and a music dataset from Last.FM., both mapped through DBpedia resources into Resource Description Framework (RDF) graphs.

1.3.1 Specific Goals

The general goal mentioned above has been broken down into 4 specific goals to be achieved in this thesis:

- SG1: Propose a feature selection approach to filter relevant properties according to the domain.
- SG2: Propose a personalization methodology that weighs links in a LOD graph based on the user's past ratings on the items in a recommender system.
- SG3: Propose a user profile modeling from the personalized graph obtained in the previous step.
- SG4: Propose methods to leverage the semantics of literals properties in LOD-based semantic similarity.

The specific goals are mentioned throughout chapters 5 and 6 whenever the steps of the developed solution are explained, to clarify which part of this work is responsible for achieving each of the objectives.

1.4 METHODOLOGY

This work investigates and proposes diverse approaches to personalize LOD-based Recommender Systems. The research protocol is composed of the following steps:

1. **Literature review**: Initially, we conducted a literature review to understand the state-of-the-art for LOD-based Recommender Systems and some common personalization methods, such as Feature Selection based on summarization of RDF properties. The literature review provides a solid background to the research, presenting different points of view about the research topics.
2. **Mapping research opportunities**: We identified two techniques to improve LOD-based RS using the knowledge obtained from the literature review. Our research suggests that combining links personalization and literal similarity would result in improved recommendations compared to baseline semantic similarity measures.
3. **Prototype implementation**: We modeled a prototype to test the proposed approach. We developed an automated personalization engine capable of accessing LOD datasets and calculating the degree of importance that each property whether it is an object property or a datatype property- has to the user, based on her past choices within the RS, regardless of the domain.

4. **Experimental evaluation**: In this step we implement and evaluate the proposed methods by comparing results using the personalized user model versus results using only the baseline methods. The experiments are implemented under two datasets: i) the first one in the movie domain and; ii) the second one in the music domain.

5. **Evaluating the results obtained**: Finally, we employ ranking evaluation metrics described in Section 3.6 to assess the quality of the results. The metrics scores are analyzed, and we discuss the results and suggestions for future improvements.

1.5 STATEMENT OF THE CONTRIBUTIONS

This research aims to contribute to the LOD-based Recommender Systems research area in multiple ways. We describe the contributions in the following:

1. **Literature review**: The fundamentals offer an exploratory study in the Semantic Web and Recommender Systems research areas. Then, we present the specific background knowledge for working with semantic LOD-based Recommender Systems. This study encompasses a critical analysis of recent research combined with an investigation of study cases on the problem discussed.

2. **Solution Proposal**: After analyzing the last studies conducted in the research area, we propose a novel approach to personalize recommendations in LOD-based Recommender Systems. We develop diverse branches of algorithms to improve the precision of existing or novel semantic similarity measures. Some algorithms are purely novel, as the Literal Similarity measure, others are a new form of applying algorithms, adapted to databases or personalization strategies, as in the PLDSD method, which is a new personalizing approach applied before the LDSD stage of the Recommender System (RS).

3. **Literal Similarity (LiSim)**: We developed a novel Literal Similarity measure that can set the degree of matching between two literal properties — such as strings or numbers. The development of this measure provides a way to leverage the neglected datatype properties, augmenting the accuracy of the system and thus improving the results. Existing LOD-based similarity methods only focus on the object properties — i.e., links to other resources in the RDF graph.

4. **Personalized Linked Data Semantic Distance (PLDSD)**: We also developed a personalization method that analyzes the graph features according to the user's preferences. This research uses SPARQL to access LOD datasets and calculate the weights of properties for a target user of the system by means of a function called W. These weights are used in future recommendations to give more importance to the properties that appear frequently in the items already evaluated by the user.

5. **Personalized Summarization**: The solution includes a preprocessing step that filters the most important features given a specific domain. As implementation of this feature selection step, we adapted the Summarization method (NOIA et al.,

2018) to the personalized model. We also have made sure to allow for the use of other feature selection methods combined to the model.

6. **Generic Architecture**: Our solution architecture presented at the beginning of Chapter 5 translates the needs of technical research into a practical solution. Our architecture is designed to allow the addition of new semantic similarity or feature selection algorithms at specific points in the implementation. Although this research has prioritised a few algorithms as the basis for experiments, future work that may continue this research could include other algorithms and databases. Thus, the solution's generic architecture establishes rules and instructions for the proper implementation and conduct of experiments

7. **Mathematical formalism**: For a better understanding of this work, we have developed a mathematical formalism that clearly and objectively describes the LOD datasets involved in user profile modeling and in the personalization stages of the algorithms (See Chapter 5). This formalism include notations to represent any dataset that follows the Linked Data principles, their resources, properties and triples that they form. The mathematical notations made it possible to mix in the same operation data collected from simple textual databases of user evaluations with Linked Data Information extracted from DBpedia, since the representation was standardized to a graph format. Thus, it was possible to formalize functions and equations and implement them later as part of the semantic similarity algorithms that run in the experiments in this work. The formalism developed in this work can contribute to the dissemination of the research and also to other scientists of any area who use LOD databases in their investigations, as they can benefit from the notations to represent their own algorithms involving resources and relationships in RDF.

8. **Data curation and code made openly available**: We used two user rating databases available on the web to conduct experiments that approximate the functioning of an online recommendation system: MoviLens and Last.fm. The subject of the research is recommendations in LOD-enriched semantic systems, so we mapped the films and artists in these two databases to DBpedia resources, making the databases semantic. We have made the complete project available under the Github URL: <https://github.com/gbrlamota/lodweb-pldsd>. The coding uses exclusively open source technologies, with the intention of being available to any researcher who wants to reuse or evolve the code.

1.6 THESIS STRUCTURE

This chapter introduces the research topic along with the motivation and possible solutions. Moreover, we expose the objectives, the applied methodology, and the expected contributions of this research. Chapters 2 and 3 discuss the literature review of the main subjects of this work. They present an overview of the fundamental concepts that guide this proposal, such as Linked Open Data (LOD) principles, Resource Description Framework (RDF), and Recommender Systems (RSs) concepts. More specifically, we describe how LOD-based Recommender Systems work and discuss some feature selection methods to reduce the matrix dimension. Chapter 4 lists classical and recent research in the two main fields approached by this work: LOD-based RS and Feature Selection. Chapter 5 introduces and describes in detail the branch strategies to personalize the user model to improve the precision of LOD-based RSs. It presents the proposed solution and the algorithms developed with this goal. We describe the experiments in Chapter 6 together with the selected datasets, the evaluation methodology, and the metrics used. Then, we detail the experiment parameters and plot the results for graphical visualization. This chapter also includes a discussion of the results of each of the experiments. Finally, Chapter 7 concludes the work, provides information about points of improvement, and suggests opportunities for future work.

Chapter 2

THE SEMANTIC WEB

This chapter aims to present concepts related to the Semantic Web. The chapter consists of the following sections: Section 2.1 introduces the chapter. Section 2.2 presents the basics of Resource Description Framework (RDF); Section 2.3 presents the SPARQL query and illustrates how to use it; Section 2.4 presents the concepts of ontologies; and Section 2.5 introduces the Linked Open Data (LOD), and Section 2.6 concludes the chapter.

2.1 INTRODUCTION

In 2001, Tim Berners-Lee stated: "*Most of the Web's content today is designed for humans to read, not for computer programs to manipulate meaningfully*" (BERNERS-LEE et al., 2001). Indeed, web applications can parse a webpage for layout and text processing. For example, it is possible to identify a header, or a link, to extract information about the content on the page. However, they have no reliable way to process the semantics. The Semantic Web brings structure to the meaningful content of web pages, enabling web applications to answer sophisticated user queries without using complex artificial intelligence solutions. The Semantic Web is an extension of the World Wide Web that improves data sharing, discovery, integration, and reuse. The Resource Description Framework (RDF) and the Web Ontology Language (OWL) are employed to achieve these goals. RDF describes knowledge graphs, while OWL expresses type logics (called *ontologies*) attached to these graphs (SARKER et al., 2017).

Along with these new data models, it arises the need for a new query language to extract the information. Since the RDF release, several query languages have been proposed (see (HUTT, 2005) for further description). In 2004, the RDF Data Access Working Group released the first draft of SPARQL — a recursive acronym for SPARQL Protocol and RDF Query Language — a query language for RDF. In essence, SPARQL is a graph-matching query language where the query consists of a pattern matched against a data source. The values obtained from this matching are processed and generate the answer to the user (PÉREZ; ARENAS; GUTIERREZ, 2009).

The advantages of SPARQL are its expressivity and its scalability for large RDF stores thanks to highly optimized SPARQL engines (e.g., Virtuoso, Jena) (FERRÉ, 2014). Query expressiveness determines the type of queries a user can pose and how complex is the evaluation of this query. SPARQL has an expressive power equivalent to Relational Algebra (ANGLES; GUTIERREZ, 2008). This proposal uses SPARQL to access LOD datasets to find the most important features for a particular user of a Recommender System.

2.2 RESOURCE DESCRIPTION FRAMEWORK (RDF)

The Resource Description Framework (RDF) is a framework for representing information on the Web. It has an abstract syntax and formal semantics that allow deductions about the RDF data. RDF represents information in a minimalist and flexible way, which is essential for sharing information between applications that have individual design configurations. This structure increases the value of information as it becomes accessible to more applications across the Internet (KLYNE; CARROLL, 2006).

The RDF structure is a collection of triples, each of them structured as a subject, a predicate, and an object. A set of such triples is called an RDF graph. Figure 2.1 illustrates an RDF graph a diagram formed of nodes and directed-edges between them. In this graph, each triple is represented as a node-edge-node link (for this reason, the term "graph" is employed) (KLYNE; CARROLL, 2006; PAN, 2009).

A Uniform Resource Identifier (URI) is employed to identify the resources described in RDF. When an RDF node has a URI label (like the gray ones in Figure 2.1), the URI identifies the resource the node represents. Consequently, RDF assumes that nodes with the same URI represent the same resource (GARSHOL, 2003).

Each triple expresses a statement of a relationship between the pair of linked nodes. Each triple has three parts:

1. a subject;
2. an object, and;
3. a predicate (also known as property) that denotes a relationship.

The nodes of an RDF graph are the subjects and objects of triples, while edges are the predicates. The edge always points toward the object. For instance, Figure 2.1 exemplifies an RDF graph describing a Person identified by <http://www.w3.org/animal/EM/contact#me> (subject), whose name is *Gabriela*, whose email address is *gabrielaoms@ufba.br*, and whose title is Msc. The predicates are the URIs near the edges (e.g., <http://www.w3.org/2000/10/swap/pim/contact#mailbox>), and the objects are the values inside the rectangles or ellipses. An object can be a literal (e.g. Gabriela), or an RDF URI reference (<http://www.w3.org/2000/10/swap/pim/contact#Person>), or a blank node.

Resource Description Framework Schema (RDFS) is the most basic schema language commonly used in the Semantic Web technology stack. In fact, many vocabularies are

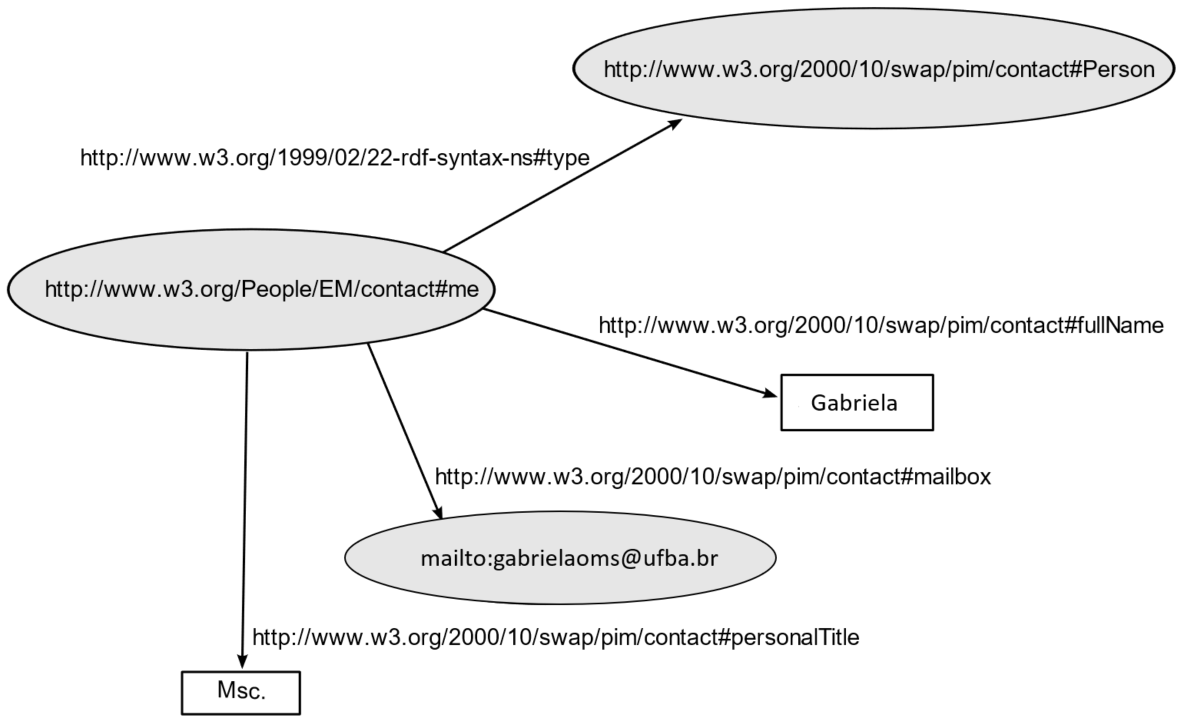

http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/EM/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName
Gabriela
http://www.w3.org/2000/10/swap/pim/contact#mailbox
mailto:gabrielaoms@ufba.br
http://www.w3.org/2000/10/swap/pim/contact#personalTitle
Msc.

Figure 2.1: Example of an RDF graph describing a person.

written in RDFS, as is in the Friend of a Friend (FOAF) vocabulary. RDFS defines the classes of the resources, the subclasses of classes, the properties of relations, the domain and range of properties, among other metadata of RDF graphs (GUO; ALAMUDUN; HAMMOND, 2016). A typical example of an rdfs:Class is foaf:Person in the Friend of a Friend (FOAF) vocabulary (BRICKLEY; MILLER, 2004). An instance of foaf:Person is a resource that is linked to the class foaf:Person using the rdf:type property. For instance, if the graph in Figure 2.1 were using FOAF, the following formal expression of the natural-language sentence: "Gabriela is a Person" could be represented by the triple: ⟨ <http://www.w3.org/animal/EM/contact#me>, rdf:type, foaf:Person ⟩, or simple ⟨ ex:Gabriela, rdf:type, foaf:Person ⟩ if we shorten the URI using an "ex" prefix.

In essence, an RDF triple denotes some relationship, indicated by the predicate, between the things denoted by the subject and the object of the triple. Subjects and objects are of certain Class, defined by an rdf:type. Predicates are of class rdf:Property, and have classes defined for their rdfs:domain and rdfs:range. For example, the following triples are used to express that the property $ex : employer$ relates a subject, which is of type foaf:Person, to an object, which is of type foaf:Organization: ⟨ ex:employer, rdfs:domain, foaf:Person ⟩ and ⟨ ex:employer, rdfs:range, foaf:Organization⟩. Given the previous two declarations, from the triple: ⟨ ex:Gabriela, ex:employer, ex:UFBA ⟩ can be inferred that ex:Gabriela is a foaf:Person, and ex:UFBA is a foaf:Organization.

According to Klyne e Carroll (2006), asserting an RDF graph is equivalent to asserting all of its triples. Therefore, the meaning of an RDF graph is the conjunction (logical AND) of the statements corresponding to all the triples it contains, along with the inferences that can be made from the relationships and metadata behind them.

2.2.1 RDF Literals

Literals are special nodes in an RDF graph that identify values such as Strings, numbers, and dates through a lexical representation. In RDFS, a Datatype Property is a special type of property that has its rdfs:range for an rdfs:Datatype instead of for an object (rdfs:Datatype is a subclass of rdfs:Literal). For example, in Figure 1.2 from Chapter 1, the dbo:foundingDate property has as its object the literal 25-01-2954 (xsd:date).

Literals may be plain or typed. A plain literal is a String combined with an optional language tag. It is considered to denote itself, so it has a fixed meaning. A typed literal is a String combined with a datatype URI (BERNERS-LEE, 2009). It denotes the member of the identified datatype's value space obtained by applying the lexical-to-value mapping to the literal String. For instance, the typed literal representations bring the data itself (25-01-2954), and also the datatype identified by URI (<http://www.w3.org/2001/XMLSchema#date>), as shown in Figure 1.2.

Some of the various existing datatypes are XML standards, while others are defined by the user/application. DBpedia, for instance, defines various datatypes to describe many of the existent units of measure, including area, currency, pressure, speed, voltage, and volume. Although, most of the existing semantic similarities discard this priceless information in the literals. Thus, one of the goals of this work is to include the semantics represented by literals in the methods for calculating similarity in LOD-based recommender systems.

2.3 SPARQL

SPARQL is a query language that can express queries across diverse data sources. The data queried using SPARQL might be stored natively as RDF or viewed as RDF via middleware. A SPARQL endpoint enables users to query a knowledge base via the SPARQL query language. DBpedia and LinkedGeoData endpoints can be accessed at <http://dbpedia.org/snorql/> and <http://linkedgeodata.org/sparql>. Listing 2.1 introduces a SPARQL query to obtain features within 200 m from the point of interest. In Listing 2.1, *objectURI* is a URI to the point of interest.

```
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?resource ?label ?location
WHERE {
objectURI geo:geometry ?sourcegeo.
?resource geo:geometry ?location;
rdfs:label ?label.
FILTER( bif:st_intersects( ?location, ?sourcegeo, 0.2 )).
}
```

Listing 2.1: SPARQL query to obtain objects within 200 m from a point of interest.

SPARQL contains capabilities for querying graph patterns along with their conjunctions and disjunctions. A SPARQL query consists of a pattern matched against a data source, and the values obtained from this matching are processed to answer. The results of SPARQL queries can be result sets or RDF graphs. Listing 2.2 introduces a SPARQL query to obtain objects within a 20 km radius of New York City.

```
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT DISTINCT ?resource ?label ?location
WHERE {
dbr:New_York_City geo:geometry ?sourcegeo.
?resource geo:geometry ?location;
rdfs:label ?label.
FILTER( bif:st_intersects( ?location, ?sourcegeo, 20 )).
}
```

Listing 2.2: SPARQL query to obtain objects within 20 km radius of New York city.

The predicate *geo:geometry* is defined at Geo-SPARQL (PERRY; HERRING, 2012), an ontology that represents features and geometries. In Listing 2.1, the variable *location* matches the spatial coordinates of objects around a point of interest. The function *bif:st_intersects()* returns true if there is at least one point in common between the spatial coordinates *location* and *sourcegeo*. The tolerance for the matching in linear distance units is supplied at the third parameter of *bif:st_intersects()*. The tolerance is 200 m as illustrated at Listing 2.1.

2.4 ONTOLOGIES

Ontology provides a foundation for the common understanding of some areas of interest among people. Even if the people do not know each other or have different traditions and languages, the ontology may be enough to make them understand each other (DIETZ, 2006). In other words, an ontology is a formal specification of a shared conceptualization (GRUBER, 1995). *Conceptualization* stands for the concept meaning and its relationships in a domain of knowledge, while *specification* stands for the formal, declarative, and explicit definition of this concept and its relationships.

The Web Ontology Language (OWL) is used in the Semantic Web to formally describe relationships between concepts. In effect, machines and humans can understand ontologies represented by OWL. Ontologies provide a common concept structure where shareable and reusable LOD datasets are built. Therefore, ontologies facilitate interoperability and data incorporation. In addition, OWL enables applications to make precise inferences like class or instance inferences without requiring the description of all concept relationships.

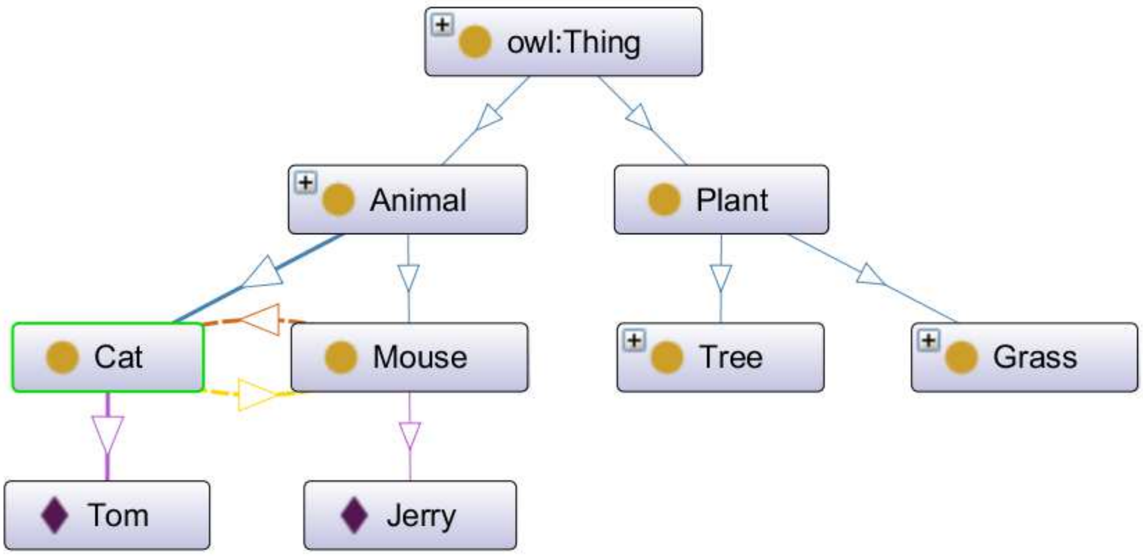


Figure 2.2: An ontology graph representing concepts and relationships between concepts.

Ontology classifies things in terms of semantics or meaning. OWL achieves this through classes, subclasses, and instances (individuals). Figure 2.2 illustrates an ontology graph describing classes and subclasses. Usually, the root node in the ontology graph is *owl:Thing*. In essence, every concept is a subclass of this root node. We can observe that this ontology defines *Cat* and *Mouse* as subclasses of *Animal*, and *Tree* and *Grass* as subclasses of *Plant*. The individuals are members of a given OWL class, so that we can define "Tom" as a member of the Cat class. This way, we can infer that "Tom" is an animal too because *Cat* is a subclass of *Animal* in the ontology graph.

There are two types of property in OWL to which an individual are related: i) object properties (owl:ObjectProperty) relate individuals of two OWL classes, and ii) datatype properties (owl:DatatypeProperty) relate individuals (instances) of OWL classes to literal values. For instance, it is possible to create an object property to describe that Cat eats Mouse as described in Listing 2.3 and 2.4. First, it is defined the relationship *eats* using *<owl:ObjectProperty>* (Listing 2.3), then *<owl:Class>* defines the class Cat while the *<owl:Restriction>* defines that every instance of Cat eats an instance of Mouse Listing 2.4. Therefore, it is possible to infer that Tom eats Jerry. The relationship *eats* is represented by the yellow edge, while the relationship *eaten_by* is described by the red edge in Figure 2.2.

```
<!-- http://semantic.org/people#eats -->
```

```
<owl:ObjectProperty rdf:about="http://semantic.org/animal#eats">
  <rdfs:domain rdf:resource="http://semantic.org/animal#animal"/>
  <rdfs:comment></rdfs:comment>
  <rdfs:label>eats</rdfs:label>
</owl:ObjectProperty>
```

Listing 2.3: Object property representing the relationship "eats".

```
<!-- http://semantic.org/animal#cat -->

<owl:Class rdf:about="http://semantic.org/animal#Cat">
  <rdfs:subClassOf rdf:resource="http://semantic.org/animal#Animal"/>
  <rdfs:subClassOf>
    <owl:Restriction>
        <owl:onProperty rdf:resource="http://semantic.org/animal#eats"/>
        <owl:allValuesFrom rdf:resource="http://semantic.org/animal#
           Mouse"/>
        </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:comment></rdfs:comment>
  <rdfs:label>Cat</rdfs:label>
</owl:Class>
```

Listing 2.4: Cat class definition including that Cat eats Mouse.

Similarly, it is possible to create a datatype property defining the number of legs of an animal, as described in Listing 2.5. First, it is set the property using *<owl :Datatype-Property>*; then the property is used to relate the individual *Jerry* with the literal value 4 representing his number of legs.

```
<!-- http://semantic.org/animal#legs -->

<owl:DatatypeProperty rdf:about="http://semantic.org/animal#legs">
  <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
</owl:DatatypeProperty>

<!-- http://semantic.org/animal#Jerry -->

<owl:NamedIndividual rdf:about="http://semantic.org/animal#Jerry">
  <rdf:type rdf:resource="http://semantic.org/animal#Mouse"/>
  <legs rdf:datatype="http://www.w3.org/2001/XMLSchema#integer">4</legs>
</owl:NamedIndividual>
```

Listing 2.5: Datatype property describing the number of legs in an animal.

It is important to realize that RDF defines the data structure, while OWL describes semantic relationships. RDF allows the user to link concepts together, so it is possible to describe that the contact's (concept) name is Gabriela (another concept), as described in Figure 2.1. In brief, the triples describe a single fact: "contact#me fullName Gabriela". However, it is not possible to classify objects using RDF. For example, it is impossible to infer that a Person is a subclass of human-being.

OWL is a more expressive knowledge representation than RDF. It categorizes properties (relationships) into object and datatype properties, enabling the user to add restrictions on properties. For example, it is possible to define *Tom* as a Cat and infer that *Tom* eats *Jerry* because every Cat eats every Mouse in the Class definition. This information is not possible to obtain using RDF.

OWL is in its second version (OWL 2), which extends OWL 1 to facilitate ontology development and sharing. OWL 2 has a similar overall structure to OWL 1 but adds new functionality like new constructs for properties, extended support for datatypes, and extended annotations.

2.5 LINKED OPEN DATA - LOD

The Web has evolved into a space where documents and data are linked (BIZER; HEATH; BERNERS-LEE, 2009). To support this new Web, a set of practices for publishing and connecting structured data has been proposed by Berners-Lee (BERNERS-LEE, 2011). This set of practices is known as Linked Data because it enables users to start browsing in one data source and then navigate along with links into related data sources. In addition, Linked Data is published so that the data is machine-readable, enabling new possibilities for applications. Berners-Lee (BERNERS-LEE, 2011) defines the following set of practices to create Linked Data:

1. Use Uniform Resource Identifiers (URIs) as names for things;
2. Use HTTP URIs to publish your data;
3. Provide useful information using the standards (RDF, SPARQL);
4. Include links to other URIs so that users can discover more things.

The LOD project is formed from many datasets[1] that encompass a vast collection of *statements* related to *entities* — also called *resources* — as persons, places, songs, movies, schools, and diseases, in different domains, including Social Networking, Arts, Government, and Life Sciences. Some of the LOD standards to link the data include making resources and their connections available in a common format — for example, RDF — and making resources available through a SPARQL endpoint using unique identifiers (BERNERS-LEE, 2009).

The World Wide Web Consortium (W3C) is the institution responsible for providing technology patterns, including RDF and SPARQL, for accessing, using, and manipulating the content of LOD datasets. Therefore, Linked Data relies on these three technologies:

[1]The LOD cloud diagram contains 1,314 datasets with 16,308 links, as of September 2023 (MCCRAE, 2023).

Uniform Resource Identifier (URI) (BERNERS-LEE; FIELDING; MASINTER, 2005), the HyperText Transfer Protocol (HTTP) (FIELDING et al., 1999), and the Resource Description Framework (RDF) model. A simple way to create linked data is using one RDF file with a URI that links to another. A supposed RDF file, named <http://exam ple.org/Hotels>, where hotels around the world are described. Local identifiers (Venice, Italy, and Hotel_Danieli) describe one hotel (resource). In Listing 2.6, hotel Danieli is described with RDF. An HTTP URI <http://example.org/Hotels/Hotel_Danieli> can be assigned, enabling anyone on the Web to access the hotel's description.

```
<rdf:Description about="Hotel_Danieli"
  <rdf:type rdf:Resource="Italy">
  <rdf:type rdf:Resource="Venice">
</rdf:Description>
```

Listing 2.6: Description of hotel Danieli in an RDF file.

There is another example RDF file (listing 2.7) containing the description of Hotels in Venice. Hotel Danieli is in Venice; however, describing it again in Listing 2.7 is unnecessary. Hotel Danieli is described by its HTTP URI, which points to its description. When released under an open license, these files are called Linked Open Data (LOD). This work uses two domains from the LOD source DBpedia: movie and music. These datasets are described in Chapter 6, Section 6.2.

```
<rdf:Description about="Hotels_in_Venice"
  <rdf:type rdf:Resource="http://example.org/Hotels/Hotel_Danieli">
</rdf:Description>
```

Listing 2.7: Description of hotels in Venice in a RDF file.

Tim Berners-Lee Berners-Lee (2011) suggested a 5-star rating system for Linked Open Data, illustrated in Figure 2.3. The more stars the data has, the more shareability “power" it contains. Below, we describe what is necessary to achieve each star:

- **1 star** - Data available on the Web under an open license. Even a PDF or image scan is allowed whether the information is public.
- **2 stars** - Data delivered as structured (machine-readable) data. For example, an Excel file instead of an image scan of a table.
- **3 stars** - Data available in a non-proprietary open format, like using CSV instead of Excel.
- **4 stars** - All requirements above plus using open standards from W3C (e.g., RDF and SPARQL) to identify things and properties. Following this standard, users can point their data at other data.
- **5 stars** - All requirements above plus link your data to other data to provide context.

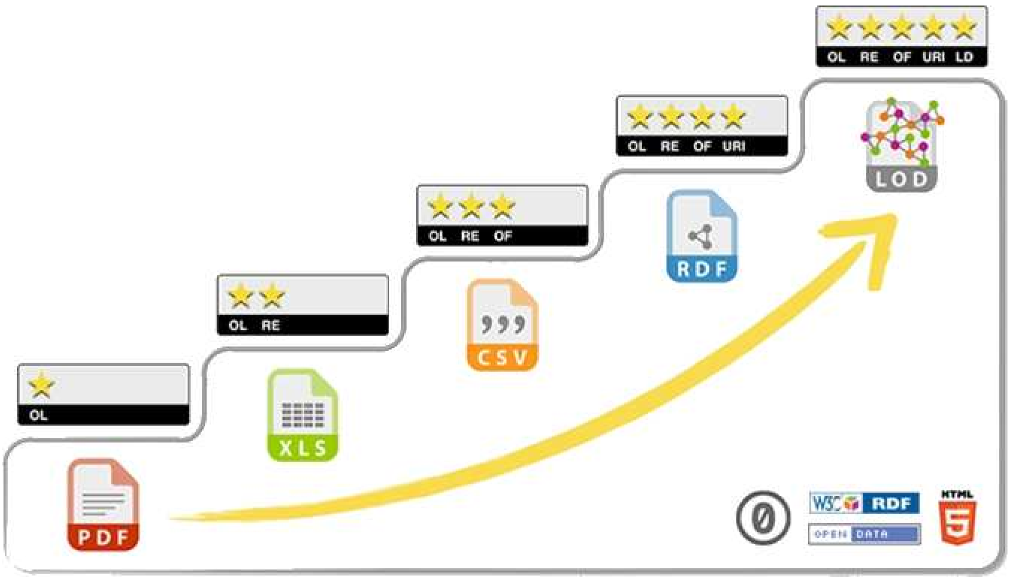


Figure 2.3: Five Star Scheme for Linked Open Data. (KIM, 2019)

A notable example of LOD usage is the *Linked Open Data (LOD) Project* that started in 2007 to offer public access to LOD datasets. In 2019, this project connected 1,239 datasets with 16,147 links between them (MCCRAE, 2023), resulting in more than 31 billion items (FRESSATO, 2019). This collection of datasets is known as the LOD cloud. As a result, web search engines can use HTTP URIs to access data within different LOD datasets, effortlessly generating new (and possibly more precise) information. Moreover, applications obtain other benefits from LOD, such as facilitating data reutilization, extension, and shareability (TRIPERINA et al., 2015).

2.5.1 DBpedia

The central node in the LOD cloud is the DBpedia dataset. It has derived its data corpus from Wikipedia, a heavily visited and under constant revision online encyclopedia. The DBpedia Association maintains the dataset and provides an HTTP service endpoint to execute queries. One must submit a query using SPARQL language to query LOD data. For this reason, the endpoint usually is called the SPARQL endpoint (LEHMANN et al., 2015). One can ask queries against DBpedia using the OpenLink Interactive SPARQL Query Builder (iSPARQL)[2], the SNORQL query explorer [3], or any other SPARQL-aware client. In this research, we use ARQ[4] to access DBpedia. ARQ is a SPARQL processor for Jena - a free, open-source framework for building Semantic Web and Linked Data applications.

DBpedia is in the Cross Domain category of the LOD cloud, shown in Figure 1.1 of Chapter 1. We can see where DBpedia is situated in Figure 2.4, which illustrates the

[2]http://dbpedia.org/isparql
[3]http://dbpedia.org/snorql
[4]https://jena.apache.org/documentation/query/service.html

cross-domain LOD sub-cloud highlighting DBpedia and the related datasets (MCCRAE, 2023).

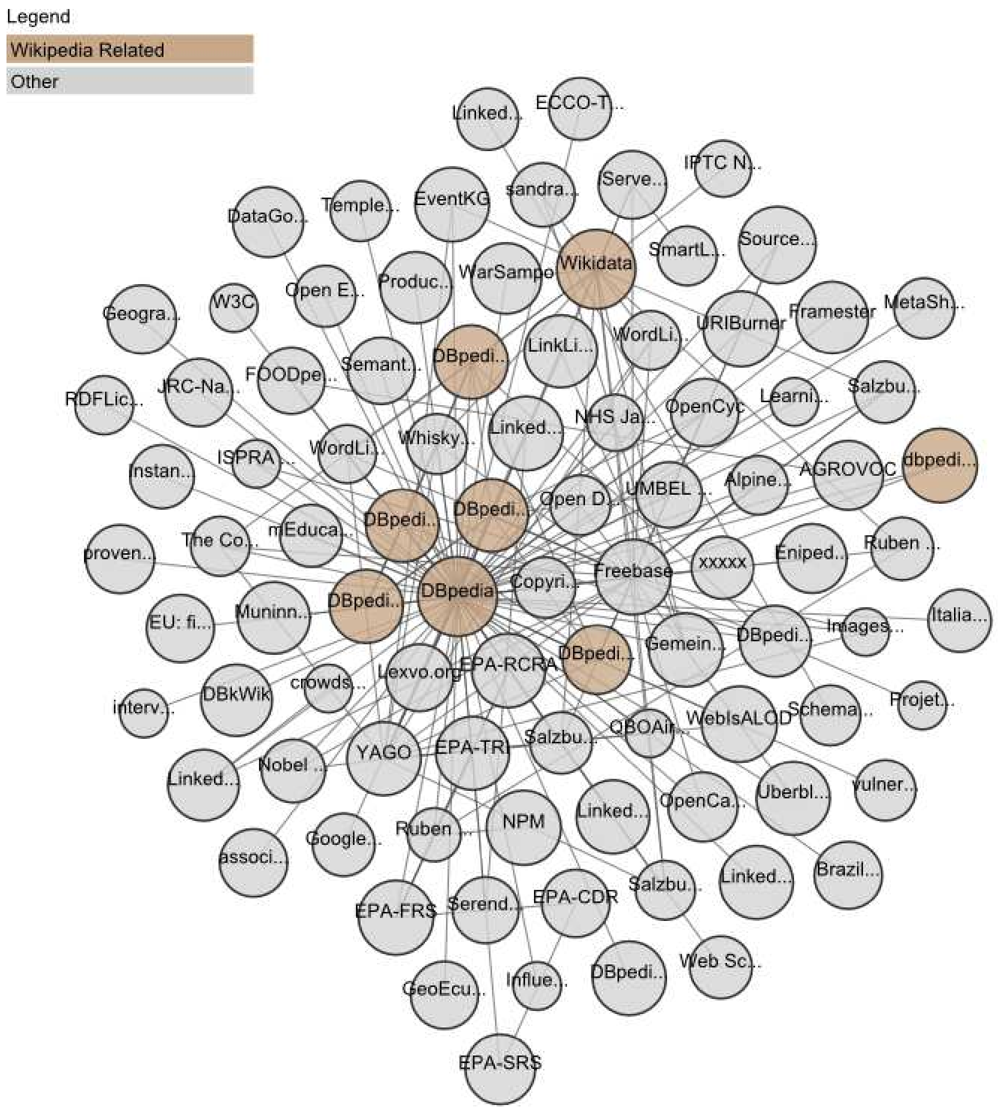


Figure 2.4: The Cross-Domain Linked Open Data Cloud from lod-cloud.net. Wikipedia-related LOD datasets are highlighted (MCCRAE, 2023)

The DBpedia ontology is the heart of DBpedia and it is continuously improved by the community contributions to the DBpedia ontology schema and the DBpedia infobox-to-ontology mappings (CONTRIBUTORS, 2022). The most recent Snapshot Release (December 2022) encompasses a total of 55 thousand properties, whereas 1,377 of these are defined by the DBpedia ontology. The English version of the DBpedia knowledge base describes 7.62 million entities, including 1.79 million persons, 7.48 million places, 610,589 creative works (music albums, films, and video games), 345,523 organizations (companies and educational institutions), 1.93 million species, 77,180 plants, and 10,591 diseases. In addition, DBpedia encompass 62 million community-contributed cross-references and owl:sameAs links to other linked data sets on the Linked Open Data (LOD) Cloud, “that allow to effectively find and retrieve further information from the largest, decentral, change-sensitive knowledge graph on earth that has formed around DBpedia since 2007” (HOLZE, 2023).

DBpedia has several advantages over existing knowledge bases, such as i) wide domain coverage; ii) information generated from real community agreements; iii) automatic evolution as Wikipedia changes, and; iv) it is multilingual (LEHMANN et al., 2015). At this point, it is necessary to clarify that the term domain is used in two forms in this thesis. The first and most constant form refers to the knowledge domain of a graph. When we said in the sentence above that “DBpedia has a wide domain coverage”, we are saying that within the DBpedia graph there are countless subgraphs on various subjects, such as personalities, places, films, music, books etc. For the experiments in this work, we curated two subgraphs, one in the Movie Domain and the other in the Music Domain.

The second form in which the term domain is employed in this work is when we are discussing an ontology in OWL or RDFS. The DBpedia ontology is built using these two metalanguages. Each DBpedia property is an instance of the rdf:Property class, which has a domain (rdfs:domain) and a range (rdfs:range) (GUHA; BRICKLEY, 2014):

- The rdfs:domain of an rdf:Property declares the class of the subject in a triple whose predicate is that property;
- The rdfs:range of an rdf:Property declares the class or datatype of the object in a triple whose predicate is that property.

In other words, in this context, the term domain refers to the rdfs:Class that a subject can assume in that triple. Back to the example in Figure 1.3, we have the following RDFS structure for the property dbo:director: ⟨ dbo:director, rdfs:domain, dbo:Film ⟩ and ⟨ dbo:director, rdfs:range, dbo:Person ⟩.

Both domain concepts intersect when we observe that a knowledge graph for a given domain has, if not all, most of its triples starting from the same domain for different ranges. For example, in DBpedia’s Movie Domain, the rdfs:domain of properties are from the dbo:Film class, as is in Figure 1.3, whose subject of the triple corresponds in RDFS to ⟨ dbr:The_Avengers, rfd:type, dbo:Film ⟩. In the experiments, the Movie Domain database has indeed all its triples starting from the OWL dbo:Film domain. However, in the Music Domain database, the triples originate from two OWL domains: dbo:MusicalArtist and dbo:Band. Therefore, we were able to include a larger number of triples in the graph. This is because the music universe in DBpedia is smaller in comparison to the Movie Domain.

2.6 SUMMARY

This chapter presented an overview of the Semantic Web. Firstly, it started by introducing the Resource Description Framework (RDF) structure in Section 2.2 and then presenting the query language SPARQL in Section 2.3. It describes how to use SPARQL to find resources and properties in a search space used as an example. The concepts of ontologies were discussed in Section 2.4. Section 2.5 presented the Linked Open Data (LOD) as one of the core concepts of the Semantic Web and also explained the DBpedia dataset, which is the semantic dataset used in this work.

Chapter 3

RECOMMENDER SYSTEMS

This chapter discusses a literature review in Recommender Systems (RSs). Firstly, we present an introduction about general RSs, followed by concepts about recommendation tasks, user modeling, recommendation techniques and recommendation problems. Then, we discuss the various types of evaluating a Recommender System, and how RSs can take advantage of using the data openly available in Linked Open Data databases and present the particularities of those types of RSs. Finally, we point the characteristics and advantages of Feature Selection and Feature Ranking tasks in a LOD-based RS.

3.1 INTRODUCTION

Always making choices and making decisions are tasks that are part of the daily life of individuals, such as buying a product, listening to music, and watching movies. However, having enough previous knowledge about several subjects in the most varied contexts, despite the many options available, is a characteristic of our society, making it difficult to find what is of real interest. Naturally, individuals search for recommendations manually at all times in several domains. Friends and/or close acquaintances are consulted to get suggestions/opinions about where to travel, which car to buy, which electronic device, which restaurant has the best food, etc.

People have different preferences and not always the suggestions/opinions of others are the same or similar to their interests. Therefore, if the recommendations come from a reliable source, the process of assisting in decision-making can be greatly improved. García et al. (2017) define the RSs as intelligent systems that have the purpose of helping individuals to find the information they need simply and efficiently. According to Ricci, Rokach & Shapira (2015), RSs are software tools and techniques that aim to solve the information overload problem by suggesting items most likely of interest to a particular user.

Recommender Systems are widely used in e-commerce sites such as Amazon.com[1],

[1]http://www.amazon.com

Mercadolivre.com[2], Ebay.com[3], among others. They are intended to help users find products that interest them according to their preferences within the range of products available (RICCI; ROKACH; SHAPIRA, 2015). The Social Web has provided new types of RSS. One of the most important is the recommendation of individuals, given that it has many unique characteristics and challenges within the broader domain of social RSs, i.e., RSs that segment the social media domain, for example, Facebook, Twitter, Instagram (GUY, 2018).

A simple way to generate recommendations to users would be to select in the system the best-selling products in the domain of electronic commerce or the most popular people in the domain of social networks, for example. In such cases, all users will receive the same recommendations, which may be useful in some contexts. However, the area of research in RSs is particularly interested in custom recommendations. Such recommendations tend to have a higher degree of satisfaction by individuals since they are generated according to the preferences of each one.

Generally, some RSs need users to provide a score or evaluate a recommended item using some evaluation method in addition to other metadata. Some of these methods consist of collecting information about user interests implicitly or explicitly. Next, we will discuss techniques covered in the literature that highlight ways to collect this information to better identify a user's preferences.

3.2 RECOMMENDATION TASKS

Recommender Systems have the main task of helping users to find information that is relevant according to their interests. But still, according to Ricci, Rokach & Shapira (2015), Aggarwal (2016) many other recommendation tasks can be defined for various purposes, for example:

- **Recommend a sequence**: The main objective is to recommend items that improve the user experience. For example, in the music context, a sequence of songs composing a playlist with various artists and music genres from the user's preference could be recommended.

- **Improve the profile**: The system should receive information from the user about his likes and dislikes. The user's contributions will improve the quality of the recommendations;

- **Recommend a bundle**: Recommend a group of related items that might be interesting to the user and also fits well together, e.g. recommend a list of attractions and hosting services to a user who is traveling to a certain destination;

- **Increase user satisfaction**: A combination of precise recommendations and a well-designed interface will increase the user's subjective evaluation of the system. This will increase the usability and likelihood of recommendations being accepted.

[2]http://www.mercadolivre.com
[3]http://www.ebay.com

- **Increase user loyalty**: A user must be loyal to a website that recognizes her as an old customer and treats her as a valuable visitor. This is a common feature in RSS as they calculate recommendations by evaluating information acquired from the user in previous interactions;
- **Annotation in context**: Given a list of items in a context, highlight some according to user preferences. For example, suggest other products related to those that the user has searched for before (clothes, books, cars, etc.) on an e-commerce website;
- **Find some good items**: A ranked list of items is retrieved, and the K most relevant items to be suggested to the user, known as Top-K items, are selected;
- **Find all good items**: In some cases, it is not sufficient to find some good items. All retrieved items are then sorted by relevance and presented to the user.

This work proposes an approach that focuses on the "find some good items" and "improve the profile" tasks since we personalize the user model and make the recommendations more suited to the user's taste.

3.3 USER MODELING

Intelligent systems that adapt their content taking into consideration the interests and personal needs of each user are expressed by the user model – *User Model*. The user model is a data structure used to capture certain characteristics about a specific user, and a user profile is the actual representation of a user model. Thus, the process for obtaining the user profile is known as *User Modeling* (PIAO; BRESLIN, 2018).

Some techniques are discussed in the literature, for example, explicit and implicit feedback. Both techniques aim to collect information about the personal preferences of each individual and thus try to model the profile of a user. User feedback is an indispensable part of most RSs, so its absence directly impacts the process of recommendation and the understanding of the user (JAWAHEER; WELLER; KOSTKOVA, 2014). In this study, it is used implicit feedback using the history of user activities based on their interactions in *microblog* Twitter.

3.3.1 Implicit feedback

The term implicit feedback encompasses evidence that users give through their natural behavior, from which their interests can be deduced. Of course, in many real-world situations, implicit feedback is much more available and requires no extra effort on the part of the user. For example, on a web page, it is easy to register users who have visited a URL or clicked on an ad. The system can treat these actions as a form of positive feedback for the items displayed. It makes sense for information about these past actions to contain highly relevant information to predict future actions (RICCI; ROKACH; SHAPIRA, 2015; REUSENS et al., 2017).

3.3.2 Explict feedback

The approach to explicit feedback as a source of information is more reliable for inferring user preferences than implicit feedback since the user is expressing his/her opinion on the recommendation generated. Typically, RSs use *ratings* – assessments to collect information about a user's interests and generally use a Likert scale[4] of N points to measure this assessment (JAWAHEER; WELLER; KOSTKOVA, 2014). For example, Amazon.com offers a 1-5 star rating system in its book catalog so that individuals can express their opinions so that their preferences can be analyzed. Unfortunately, components for collecting explicit feedback are not always available on a system.

3.4 RECOMMENDATION TECHNIQUES

Recommender Systems need to analyze information about items, users, and a particular context to suggest items that are of real interest to a specific user. As mentioned earlier, users have different interests, for example, in the context of music: some may like the pop genre, others the rock genre, etc. Therefore, for recommendations to be generated based on the preferences of each user it is necessary to use various techniques to obtain a better prediction (RICCI; ROKACH; SHAPIRA, 2015).

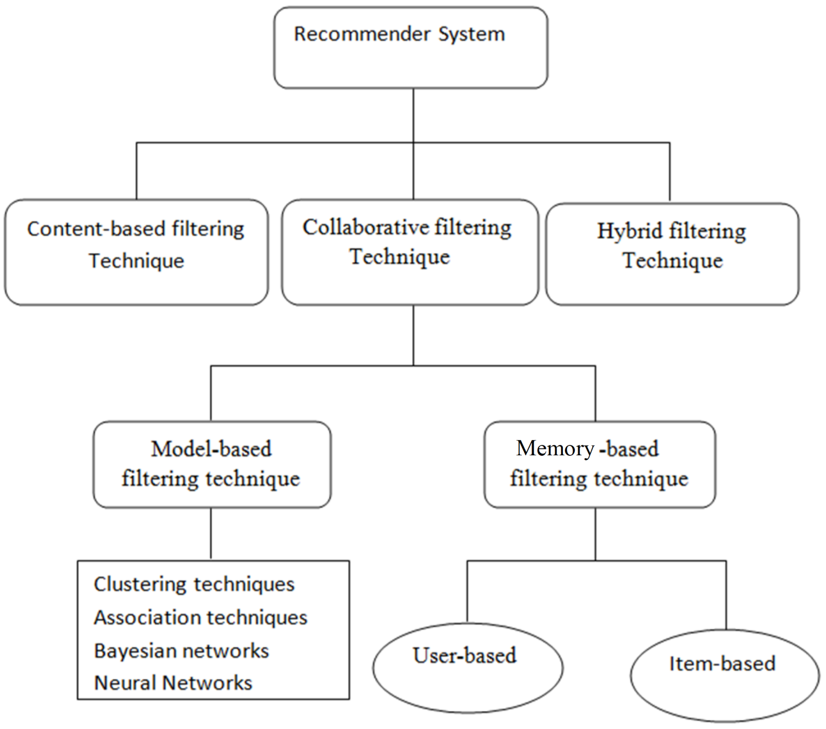

Recommender System
Content-based filtering Technique
Collaborative filtering Technique
Hybrid filtering Technique
Model-based filtering technique
Memory -based filtering technique
Clustering techniques
Association techniques
Bayesian networks
Neural Networks
User-based
Item-based

Figure 3.1: Recommendation techniques (ISINKAYE; FOLAJIMI; OJOKOH, 2015).

Several recommendation techniques were proposed in the literature as a basis for a

[4]https://en.wikipedia.org/wiki/Likert_scale

RS as illustrated in Figure 3.1. The use of them is very important so that the system can provide recommendations that are of real interest to a user, among them are Collaborative Filtering, Content-Based Filtering, Knowledge-Based Filtering, Demographic and Hybrid Filtering, which is a combination of the first two mentioned (THORAT; GOUDAR; BARVE, 2015), and will be discussed in detail below.

3.4.1 Content-Based Filtering

Naturally, individuals already make a selection of what they are only interested in, for example, they select which sections they want to read in a newspaper, the songs they like best in a *playlist* with various genres and artists, usually movies are searched by genre (action, fiction, adventure, etc.) (BARMAN; TEWARI, 2017).

Content-Based Filtering aims to analyze a set of documents and/or descriptions of items previously evaluated by a user, in order to create a model or profile of user interests based on the characteristics of the items evaluated by the user. The profile is a structured representation of user interests, adopted to recommend new relevant items. The recommendation process basically consists of combining the attributes of the user's profile with the attributes of a given item.

3.4.2 Collaborative Filtering

It's the most popular recommendation technique. Its predictions are based on evaluations or the behavior of other similar users in the system. The collaborative filtering technique works by creating a database (matrix of user items) of preferences for items by users. It then associates users with relevant interests and preferences by calculating the similarities between their profiles to generate recommendations (ISINKAYE; FOLAJIMI; OJOKOH, 2015).

In a scenario where one movie RS uses this technique, the predictions are made about the possible films that the user would watch, taking into account their evaluations submitted to previously watched films. Unlike content-based methods, without any extra information (metadata) about the users (location, genre) or items (type, category, size) collaborative filtering can build personalized recommendations (RICCI; ROKACH; SHAPIRA, 2015). The fundamental assumption behind this method is that the opinions of other users can be selected and aggregated to provide a forecast of the preference of active users (TERÁN; MENSAH; ESTORELLI, 2018).

Figure 3.2 illustrates the diagram of the Collaborative Filtering scheme. This algorithm represents the input data of the user's items as a matrix $m \times n$ of evaluations denoted as Λ. Each $a_{i,j}$ entry in Λ represents the score of the user's preference i with the j item. As well as each individual rating is within a numerical scale and can be 0 indicating that the user has not rated that item.

Table 3.1 illustrates an example of a matrix *User* $\times$ *Item* in a movie context. In this example, the user *Paul* rated the movies *Titanic* and *The Avengers* at 5 and 2 respectively. However, the movie *Hard to Kill 4.0* was not watched and it is necessary to estimate the favorable opinion of *Paul* about it. Verifying the similarity with the other

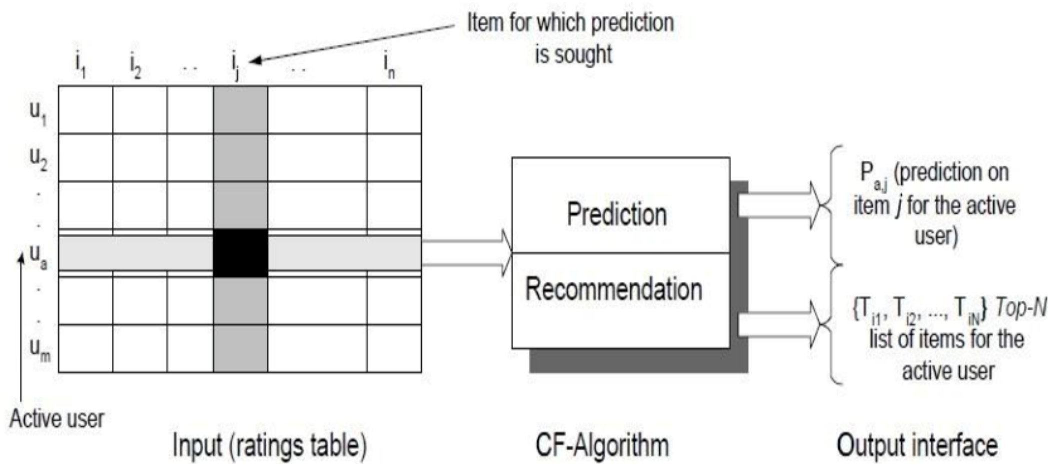

Item for which prediction is sought
i_1 i_2 .. i_j .. i_n
u_1 u_2 . . u_a . . u_m
Active user
Input (ratings table)
Prediction
Recommendation
CF-Algorithm
$P_{a,j}$ (prediction on item j for the active user)
$\{T_{i1}, T_{i2}, ..., T_{iN}\}$ Top-N list of items for the active user
Output interface

Figure 3.2: Collaborative filtering process (SARWAR et al., 2001).

users, *Fred* is the profile that most resemble that of *Paul*. Therefore, it can be noted that the assessments of *Titanic* and *Hard to Kill 4.0* follow the same pattern, demonstrating that people who liked the former may also like the latter (LÜ et al., 2012; THORAT; GOUDAR; BARVE, 2015).

Table 3.1: Users × Itens Evaluation Matrix.

Users / Movies	Fred	Joe	Paul
Titanic	5	1	5
The Avengers	1	5	2
Hard to Kill 4.0	4	2	?

There are two approaches that this technique adopts as a way to improve its predictions, which are *Memory-Based* which is based on data similarity (*user-item*); and *Model-Based*, which uses machine learning techniques to adjust a parameterized model as illustrated in Figure 3.1 (KLUVER; EKSTRAND; KONSTAN, 2018). In collaborative memory-based filtering, the *user-item* evaluations stored in the system are used directly to predict new items. This can be done in two known ways: user-based recommendation or item-based recommendation (BURKE, 2002; RICCI; ROKACH; SHAPIRA, 2015), which are discussed below.

3.4.2.1 Memory-based When using the memory-based approach we can focus on the user or on the items to make recommendations.

- **User-based**: This technique attempts to estimate a user's interest in a particular u item, aiming to calculate the similarity between a set of users (known as neighbors) N_u similar to u. This similarity is given by comparing user ratings for the same item. As well, it aims to calculate the predicted assessment for an item given to an active user, as a weighted average of the item assessments by users similar to the active user, in which the weights are the similarities of these users with the item of probable interest (RICCI; ROKACH; SHAPIRA, 2015; ISINKAYE; FOLAJIMI; OJOKOH, 2015);

- **Based on the item**: This approach calculates predictions using the similarity between items and not the similarity between users. This technique creates an item similarity model by retrieving all items that were assessed by an active user from the item matrix and determines how the retrieved items are similar to the item of interest, then it is by selecting the k most similar items and their corresponding similarities are also determined (RICCI; ROKACH; SHAPIRA, 2015; ISINKAYE; FOLAJIMI; OJOKOH, 2015).

There are several models proposed in the literature to perform the calculation of similarity between users. According to Adomavicius, Tuzhilin e Alexander (2005), the most popular metrics to perform this calculation are the Pearson Correlation and Cosine Similarity equations that will be discussed below:

- **Pearson's Correlation**: In equation 3.1, $sim(u1, u2)$ denotes the similarity between two users $u1$ and $u2$, $r_{u1,i}$ is the valuation assigned to the item i by the user $u1$, $\overline{r_{u1}}$ is the average valuation given by the user $u1$ while n is the total number of items in the user-item space. In addition, the prediction for an item is made from the weighted combination of the selected neighbor's assessments, which is calculated as the weighted deviation of the neighbor's mean values (SARWAR et al., 2001; ISINKAYE; FOLAJIMI; OJOKOH, 2015).

$$sim(u1,u2) = \frac{\sum_{i=1}^{n}(r_{u1,i} - \overline{r_{u1}})(r_{u2,i} - \overline{r_{u2}})}{\sqrt{\sum_{i=1}^{n}(r_{u1,i} - \overline{r_{u1}})^2}\sqrt{\sum_{i=1}^{n}(r_{u2,i} - \overline{r_{u2}})^2}} \tag{3.1}$$

- **Cosine Similarity**: This equation is different from Pearson's correlation in that it is a vector space model based on linear algebra and not on the statistical approach. This similarity metric calculates the angle between two $n-dimensional$ vectors ($u1$ and $u2$), and it is widely used in the fields of information retrieval and text mining in order to compare two documents, in which case the documents are represented as term vectors (ISINKAYE; FOLAJIMI; OJOKOH, 2015). The similarity between two users $u1$ and $u2$ can be defined according to Equation 3.2.

$$sim(u1,u2) = cos(\vec{u1},\vec{u2}) = \frac{\vec{u1}.\vec{u2}}{\left\|\vec{u1}\right\|_2 \times \left\|\vec{u2}\right\|_2} = \frac{\sum_{i=1}^{n} r_{u1,i} r_{u2,i}}{\sqrt{\sum_{i=1}^{n} r_{u1,i}^2}\sqrt{\sum_{i=1}^{n} r_{u2,i}^2}} \tag{3.2}$$

3.4.2.2 Model-Based This technique uses the previous assessments to learn a model in order to improve the performance of the collaborative filtering technique. The model-building process can be done using either machine learning or data mining techniques. These techniques can quickly recommend a set of items because they use a preprocessing model and have proven to produce recommendation results similar to neighborhood-based recommendation techniques (ISINKAYE; FOLAJIMI; OJOKOH, 2015). The model-based technique is not explored in this work.

3.4.3 Hybrid Filtering

The hybrid filtering technique combines a combination of two or more recommendation techniques in order to achieve better performance than Collaborative Filtering and Content-Based Filtering. This combination can be achieved in different ways in order to produce multiple outputs (KUMAR; THAKUR, 2018). According to Burke (2002) collaborative filtering is often combined with some other technique in an attempt to avoid the cold-start problem.

The Hybrid Filtering technique can help solve some of the problems associated with Collaborative Filtering and Content-Based recommenders, for example. However, regardless of type, every recommender technique will always have to deal with the cold-start problem — which will be discussed in the following Section 3.5 — since it requires a previously built set of evaluation data. Although, hybrid techniques are very popular because, in many situations, these evaluations already exist or can be inferred from data.

3.5 RECOMMENDATION PROBLEMS

3.5.1 The Cold-Start Problem

The cold-start problem is one of the main recognized problems involving RSs. Given the large number of online platforms publishing hundreds or thousands of new items every day, effective recommendation is essential for such platforms in order to keep their users continuously more engaged.

For an RS to have a good ability to suggest accurate items to a user, it is necessary to have previous information about what is of interest to them. Therefore, when new users and or items are registered in a system it might have no evaluation — implicit/explicit feedback — already stored in it. Thus, the accuracy of recommendations for these users/items is affected. It happens, for instance, when a new user has never given ratings to any item and a new item has never been rated by any user before.

Several methods have been proposed in the literature to deal with the cold start problem, mainly regarding Collaborative Filtering RS (LAM et al., 2008; ZHANG et al., 2010; GUO, 2012). According to Ricci, Rokach e Shapira (2015), Saveski e Mantrach (2014), Sedhain et al. (2014), Kumar e Thakur (2018) the cold-start problem is related to how to deal with the sparseness in the data matrix $User \times Item$ and can be identified by three different types: i) recommendation for new users; ii) recommendation for new items; iii) recommendation of new items for new users.

In the literature, there are techniques proposed as a solution to this problem, known as matrix factorization, which is mentioned in the following subsection. Other studies use Linked Open Data datasets to add semantics to the system to help address the cold-start problem (NOIA et al., 2012; GEMMIS et al., 2015; JOSEPH; JIANG, 2019).

3.5.2 The Matrix Sparsity Problem

In numerical analysis and scientific computing, a sparse matrix is a matrix in which most of the elements are zero. Large sparse matrices often appear in scientific or engineering applications, especially when solving machine learning problems. Conceptually, sparsity corresponds to systems with few pairwise interactions.

In a recommender system, the rows and columns of the matrix are the users and the items; and the element in each position corresponds to whether or not the user gave a rating to the item — if it is a binary representation — or, sometimes, it is the rating itself. This matrix is often called the *User* × *Item* matrix. Thus, the matrix sparsity problem arises from the phenomenon that users, in general, rate only a limited number of items (GUO, 2012).

Especially in Collaborative Filtering (CF), as the dimensionality of data grows, the matrix becomes more sparse, that is, there are many missing data not evaluated by the user that needs to be predicted. In the literature, there are several proposed methods that address the high dimensionality and sparseness of data, such as Matrix Factorization (RICCI; ROKACH; SHAPIRA, 2015). Another approach to resolve this issue in CF recommender systems is to address the problem through the user modeling task. Guo (2012) proposes a method that utilizes trust to find more similar users whose ratings can be aggregated to generate recommendations.

Another solution that addresses the *User* × *Item* matrix sparsity problem is using semantic data from other datasets, such as LOD datasets. These systems are called semantic aware Recommender Systems. Codina, Ricci e Ceccaroni (2013) describe an approach based on the intuition in which not only the ratings provided by the users in a specific situation can be considered as relevant, but also the ratings provided in similar situations.

3.6 EVALUATION OF RECOMMENDER SYSTEMS

Evaluating the quality of an RS is important to verify its ability to accurately predict user choices. In many applications, people use an RS for more than exact anticipation of their interests. Users may also be interested in discovering new items, quickly exploring multiple items, preserving their privacy, quick system responses, and many other properties of the recommendation engine interaction. Therefore, it is essential to identify the properties that can influence the success of an RS in the context of a specific application.

Several evaluation metrics are available in the literature, and choosing among them depends on the recommendation technique used. Another aspect that influences the evaluation metrics is the type of experiments, which are offline, online, or based on user studies (SHANI; GUNAWARDANA, 2011; RICCI; ROKACH; SHAPIRA, 2015). These types are discussed below:

- **Offline**: An offline experiment is performed using a set of collected data from users who have chosen or evaluated items. Using this data set as a start, the behavior of these users is then simulated. It is assumed that the behavior of users during the experiments is similar enough to the behavior when the data was collected so that reliable decisions are made based on the simulation;

- **User studies**: A user study is conducted by recruiting a set of test subjects and asking them to perform various tasks that require an interaction over the RS. While the tasks are performed, their behaviors are observed and recorded by collecting quantitative data. For example, which part of the task was completed, the accuracy of the task results, or the time spent to perform it. In many cases, it is necessary to ask qualitative questions before, during, and/or after the completion of the task. These questions can collect data that is not directly observable. For example, how much the individual liked the interface or whether the task was easy or difficult to complete;

- **Online**: In an online experiment, the RS tries to influence the user's behavior. They also aim to collect information about their behavior, such as: whether or not they liked a movie when evaluating it according to a numerical scale and if they were interested in similar products based on what they bought previously, among others. Thus, an online experiment has the possibility and at the same time the challenge of building the user's model based on their behavior and evaluations.

In this work, we perform offline experiments that are described in Chapter 5.

After selecting the user tasks to be supported by a system and implementing the chosen type(s) of experimentation it is necessary to perform repeatable evaluations in order to measure the recommender system utility. According to Herlocker et al. (2004), the metrics that evaluate RSs can be broadly classified into the following categories: predictive accuracy metrics, classification accuracy metrics, and ranking metrics.

3.6.1 Predictive Accuracy Metrics

Predictive accuracy metrics measure how close a user's predicted ratings are to the true user's ratings. These metrics are very important and can be used to measure the ability of an RS to evaluate items related to user preferences (HERLOCKER et al., 2004).

- **Root Mean Square Error (RMSE)**: This metric became very popular as it was used as the standard metric for the Netflix Prize[5]. A feature of RMSE is that it tends to disproportionately penalize large errors because of the term squared within the summation (AGGARWAL, 2016). It is be given by Equation 3.3 below,

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(r_i - p_i)^2}{n}} \tag{3.3}$$

[5]The Netflix Prize was an open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings. See more at <https://www.netflixprize.com/>.

where n is the total number of ratings on all users, p_i is the expected valuation for the user i in the item p, and r_i is the actual valuation again. RMSE amplifies the contributions of absolute errors between forecasts and actual values;

- **Mean Absolute Error (MAE)**:

  This metric, on the other hand, does not disproportionately penalize larger errors and aims to average the absolute difference between predictions and true rankings. It is given by Equation 3.4 below,

$$MAE = \frac{\sum_i^n \mid r_i - p_i \mid}{n} \tag{3.4}$$

  where n is the total number of ratings on all users, p_i is the expected rating for the user i on the p item, and r_i is the actual rating. The lower the MAE, the better the prediction.

3.6.2 Classification Accuracy Metrics

In many applications, the RS does not predict the ratings a user would give to items, such as movie ratings. The system sometimes tries to recommend items that the users can like. For example, when a user selects a movie, Netflix suggests a set of other movies that can also be interesting according to the selected movie. In this case, we are not interested in knowing if the system correctly predicts the ratings of these films, but if the system correctly predicts whether the user would select these films, that is if these suggested items are really of her interest (RICCI; ROKACH; SHAPIRA, 2015).

Classification accuracy metrics measure how often an RS makes correct or incorrect decisions about whether or not an item is good. These metrics are, therefore, appropriate for tasks such as "find good items" when users have true binary preferences (HERLOCKER et al., 2004). The following Table 3.2 illustrates the possible result of a recommendation of an item to a user.

Table 3.2: Classification of the possible outcome of a recommendation.

	Recommended	**Not recommended**
Of Interest	True Positive (TP)	False Negative (FN)
No Interest	False Positive (FP)	True Negative (TN)

Some of the most known classification accuracy metrics are presented below.

- **Precision**: It consists in calculating the proportion of positive examples correctly classified among all those predicted as positive. It is shown in Equation 3.5.

$$Precision = \frac{\#TP}{\#TP + \#FP} \tag{3.5}$$

- **Recall**: Corresponds to the hit rate in the positive class, as shown in Equation 3.6.

$$Recall = \frac{\#TP}{\#TP + \#FN} \tag{3.6}$$

- **F1**: This metric combines the two previous metrics, which is their harmonic mean, and is shown in Equation 3.7 below.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3.7}$$

3.6.3 Ranking Metrics

In this section, we discuss the ranking metrics, which have the purpose of measuring in which position are the recommendations considered relevant within a ranked list presented to users. Below, we show some of these metrics.

- **Mean Reciprocal Rank (MRR)**: The MRR metric is intended to calculate the accuracy of a recommended item of a ranked list of items for one user. The MRR is the average of the Reciprocal Rank (RR) in all queries for each user and is a particularly important measure for domains that generally provide users with only a few but valuable recommendations. For example, referrals from friends on social networks where of the top 3 or 5 recommendations are most important (SHI et al., 2012). This metric is given in Equation 3.8:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{3.8}$$

  where $|Q|$ is the number of queries and $rank_i$ is the position of the first relevant item in query $|Q|$ of candidate items;

- **Precision at n (P@n)**: The number of recommended items in a list for a user can vary depending on the method of recommendation used and on the amount of data available. Thus, for a number of recommended items in a ranked list to be evaluated, the $P@n$ metric presented in Equation 3.9 uses n as the cut-off point and considers the n first retrieved items, called Top-N recommendations (DAVOODI; KIANMEHR; AFSHARCHI, 2013).

$$P@n = \frac{r}{n} \tag{3.9}$$

  where n is the quantity of items returned and r is the quantity of items considered relevant up to the n position of the list;

- **Mean Average Precision (MAP)**: The MAP metric is responsible for generating a single value, which is obtained by averaging the average precision of each user's list of recommendations (PARRA; SAHEBI, 2013). It is presented in Equation 3.10, as follows:

$$MAP = \sum_{n=1}^{N} \frac{AveP(n)}{N} \tag{3.10}$$

where $AveP(n)$ is the average precision of the user n, i.e., the average of the precision values obtained given a set of Top-N recommendations (P@n) after each relevant recommendation is retrieved (MANNING; RAGHAVAN; SCHÜTZE, 2008);

- **Normalized Discounted Cumulative Gain (NDCG)**:

  NDCG is a metric based on the notion that items in a rank have varying degrees of relevance. Besides, it considers not only that notion, but once relevant items in a certain rank position may also be less valuable to the users than a less relevant one in a greater position (JäRVELIN; KEKäLäINEN, 2002). Thereby, the gained value (without discounts) is obtained as the relevance score of each item is progressively summed from the rank position 1 to n. Discounted Cumulative Gain (DCG) is then obtained through Equation 3.11, which penalizes highly relevant documents appearing lower on a search result list as the graded relevance value is logarithmically reduced proportionally to the result's position — sharper or smoother discounts can be computed to model the user's behavior by varying the base of the logarithm, b.

  $$DCG[i] = \begin{cases} CG[i], & \texttt{if } i < b \\ DCG[i-1] + G[i]/\log_b i, & \texttt{if } i \geq b. \end{cases} \tag{3.11}$$

  DCG values can be compared to the theoretically best possible score vector. This ideal vector is represented by Equation 3.12 for relevance scores 0 and 1, where m is the number of relevant items.

  $$BV[i] = \begin{cases} 1, & \texttt{if } i \leq m \\ 0, & \texttt{otherwise.} \end{cases} \tag{3.12}$$

  In order to compare the two techniques being evaluated using DCG, we need to normalize their values. The DCG vectors are normalized by dividing them by the corresponding ideal DCG vectors (iDCG). The NDCG is then calculated using Equation 3.13, which, likewise Precision@K, only evaluates the top k results (MANNING; RAGHAVAN; SCHHüTZE, 2008).

  $$NDCG(k) = \frac{DCG(k)}{iDCG(k)} \tag{3.13}$$

This work uses the following ranking metrics to evaluate our RS approach: P@n, MAP, and NDCG.

3.7 LOD-BASED RECOMMENDER SYSTEMS

The data that describe items in a Recommender System can come from many sources, most usually private databases belonging to the company that owns a website or service. Nevertheless, the availability of open knowledge sources is growing, increasing the emergence of semantics-based applications, for example, Semantics-Aware RSs (GEMMIS et al., 2015). Linked Open Data (LOD) is a widely known project that aims to connect open datasets through the Web and to facilitate their use by applications (BIZER; HEATH; Berners-Lee, 2009).

The LOD project is formed from many datasets encompassing a vast collection of such as persons, places, songs, movies, schools, and different domains. These entities are connected to others by links, making the datasets interconnected with the whole LOD diagram. This characteristic makes LOD datasets a natural source of information that can enrich diverse recommender systems engines, for example, when systems leverage information from the DBpedia dataset — already presented in Chapter 2.

Figure 3.3 shows a LOD sub-cloud with the DBpedia dataset centered. DBpedia is a multipurpose project that aims to make Wikipedia content openly available in Resource Description Framework (RDF) format and also incorporates links to other datasets on the Web, such as FOAF[6], Geonames[7], MusicBrainz[8] and BBC datasets, for instance, BBC Programmes[9]. As these extra links are provided — in terms of RDF statements — applications may exploit knowledge from interconnected datasets, thus facilitating various possibilities for developing semantic applications. For this reason, DBpedia is often considered the linking hub of the LOD project. For example, in Kobilarov et al. (2009), the authors describe how Linked Data technologies were applied within the British Broadcasting Corporation (BBC) and how DBpedia and MusicBrainz are used in that process as both are interlinking vocabulary and a data provider.

One example of a semantics-aware application is when a Knowledge Graph (KG) is used to provide better recommendations to the users (CATHERINE; COHEN, 2016). In most of the works that follow this line of research, authors enrich the content of benchmark databases with semantics given from interconnections between nodes on the KG. Another example is found in Musto et al. (2016), which studies the impact of using knowledge from the LOD cloud on the overall performance of a graph-based recommendation algorithm. They use a uniform formalism to represent both collaborative and content-based features and then investigate whether the integration of LOD-based features improves the algorithm's effectiveness and to what extent the choice of different feature selection techniques influences its performance in terms of accuracy and diversity.

This work focuses on Content-Based Recommender Systems, which compute the similarity between items in the system by comparing their characteristics (also called features). For instance, in a movie RS, algorithms reason about the degree of similarity between two movies by comparing their directors, actors, the main subject, and so on.

[6]http://xmlns.com/foaf/spec/
[7]https://www.geonames.org/
[8]https://musicbrainz.org/
[9]https://lod-cloud.net/dataset/bbc-programmes

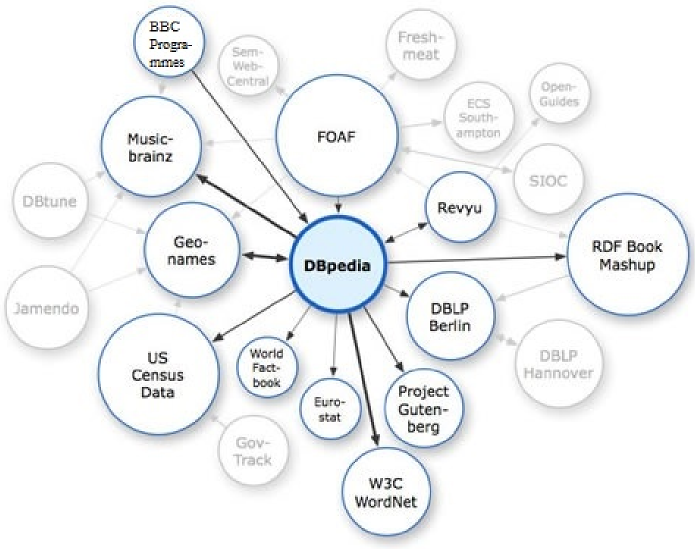


Figure 3.3: LOD sub-cloud showing some of the highly used DBpedia-related datasets (HOLZE, 2023).

The more a pair of movies share features, the greater their similarity. Figure 3.4 shows an example of relationships between resources in a music recommender system. We can see that both *Ariana_Grande* and *Selena_Gomez* are *musicalguests* of *List_of_the_-tonight_show_with_Jay_Leno_episodes*, and also they both are *subject* of the category *21st-century_American_singers*. This could indicate that these two LOD resources are similar.

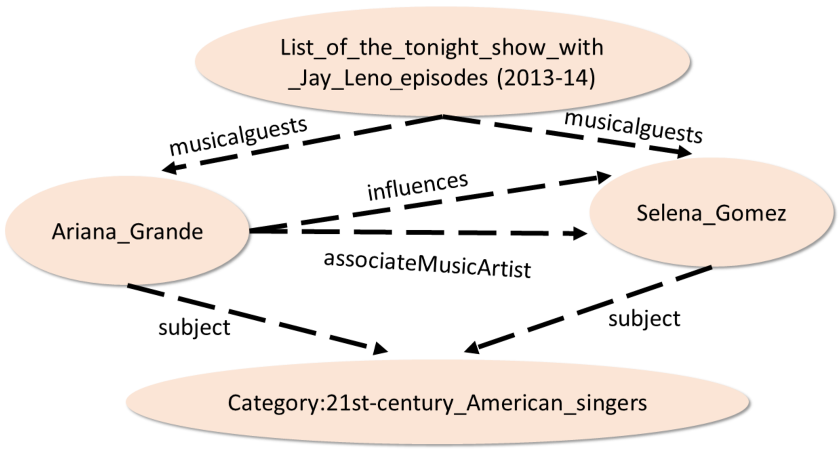

List_of_the_tonight_show_with _Jay_Leno_episodes (2013-14)
musicalguests
musicalguests
influences
Ariana_Grande
Selena_Gomez
associateMusicArtist
subject
subject
Category:21st-century_American_singers

Figure 3.4: Example of relationships in a semantic-aware recommender system. Adapted from Piao, Ara e Breslin (2016).

Although several studies address different ways to build recommendations in LOD-based environments, researchers are still discussing some problems, as shown in Section

3.5. For example, the sparsity problem, when there are feedback data in comparison to the whole data matrix size, or the cold-start problem, which happens when a new item or a new user is added to the system, making it hard to provide personalized recommendations, since interactions are not present yet.

3.7.1 Semantic Similarity Measures

LOD datasets are often used to add semantics to the system to help address those well-known RS problems (NOIA et al., 2012; GEMMIS et al., 2015; JOSEPH; JIANG, 2019). Therefore, these enriched systems use a semantic similarity algorithm that calculates the degree of matching between pairs of Linked Data resources. Because RDF represents data as a graph, these algorithms, in general, count the number of direct and indirect links — i.e., the edges in the graph —, the length of the path between two resources, or their place in the hierarchy of classes (PASSANT, 2010; PIAO; ARA; BRESLIN, 2016; CHENIKI et al., 2016). We discuss some of the existing semantic similarity measures in the next subsections.

3.7.1.1 Linked Data Semantic Distance (LDSD) LDSD is one of the pioneers' approaches for measuring the semantic distance between two resources on LOD datasets, such as DBpedia (PASSANT, 2010). The complete LDSD function is shown in Equation 3.14, and it is composed of four smaller functions, $C(p_j, r_a, r_b)$, that computes whether there is a link p_j between resources r_a and r_b. They return 1 in case there is 0 otherwise. Whenever there is an n, the function calculates the total number of links between a resource r_a or r_b to all other resources.

$$sim_{LDSD}(r_a, r_b) = \frac{1}{1 + \sum_j \frac{C_d(p_j,r_a,r_b)}{1+\log C_d(p_j,r_a,n)} + \sum_j \frac{C_d(p_j,r_b,r_a)}{1+\log C_d(p_j,r_b,n)} + \sum_j \frac{C_{ii}(p_j,r_a,r_b)}{1+\log C_{ii}(p_j,r_a,n)} + \sum_j \frac{C_{io}(p_j,r_a,r_b)}{1+\log C_{io}(p_j,r_a,n)}} \tag{3.14}$$

The function C_d only considers direct links from resource r_a to resource r_b — the first function of the equation. Once the graph is directed, reverse links, from resource r_b to resource r_a, count as different links — second function of the equation. For example, the triple ⟨*Ariana_Grande*, *influences*, *Selena_Gomez*⟩, taken from Figure 3.4, represents a direct link *influences* between the resources *Ariana_Grande* and *Selena_Gomez*. The inverse situation does not exist in Figure 3.4, since there is not a link *influences* between the resources *Selena_Gomez* and *Ariana_Grande*.

The function C_{ii} represents indirect incoming links and returns 1 only if there is a resource r_c that satisfies both $\langle p_j, r_a, r_c\rangle$ and $\langle p_j, r_b, r_c\rangle$, 0 if not — third function of the equation. It is exemplified in Figure 3.4 by the triples ⟨*Ariana_Grande*, *subject*, *Category:21st-century_American_singers*⟩, and ⟨*Selena_Gomez*, *subject*, *Category:21st-century_American_singers*⟩. In this example, a virtual (indirect) link between *Ari-*

ana_Grande and *Selena_Gomez* is created through the *subject* links incoming to the *Category:21st-century_American_singers* resource.

Finally, function C_{io}, in its turn, represents indirect outgoing links and equals to 1 only if there is a resource r_c that satisfies both $\langle p_j, r_c, r_a \rangle$ and $\langle p_j, r_c, r_b \rangle$, 0 if not — fourth and last function of the equation. It is exemplified in Figure 3.4 by the triples ⟨*List_-of_the_tonight_show_with_Jay_Leno_episodes*, *musicalguests*, *Ariana_Grande*⟩, and ⟨*List_of_the_tonight_show_with_Jay_Leno_episodes*, *musicalguests*, *Selena_Gomez*⟩. In this example, a virtual (indirect) link between *Ariana_Grande* and *Selena_Gomez* is created through *musicalguests* links outgoing from *List_of_the_tonight_show_with_-Jay_Leno_episodes.*

3.7.1.2 Vector Space Model (VSM) VSM is also employed to calculate the similarity of two concepts/resources in the Linked Data (NOIA et al., 2012). The approach represents the RDF graph as a 3-dimensional matrix. Each matrix slice represents an ontology property p. A cell in the matrix is not null only if there is a relation, through p, from a subject (on the rows) to an object (on the columns). As they use the movie domain, each movie is seen as a vector, and their components correspond to a version of TF-IDF in which the terms are resources, and the documents are movies. The similarity between the two movies is then the correlation between their vectors. The cosine of the angle between them quantifies this similarity. All nodes of the graph are represented on both rows and columns.

Properties are considered independent, and the similarity is only calculated for resources of the same type. The bigger matrix is divided into smaller ones, each corresponding to a property, having their domain as the rows and range as the columns. The weights are computed using TF-IDF. The TF($f_{n,i,p}$) part is the frequency of a node n as being the object of an RDF triple having p as the property and the movie i as the subject. The values it can assume are 0 or 1. The IDF part is the logarithm of the ratio between M, the total number of movies in the collection, and $a_{n,p}$, the number of movies linked to n by means of p. The similarity is then calculated using Equation 3.15:

$$sim_{DiNoia}(C_i, C_j) = \frac{\sum_{n=1}^{t} w_{n,i,p} . w_{n,j,p}}{\sqrt{\sum_{n=1}^{t} w_{n,i,p}^2} . \sqrt{\sum_{n=1}^{t} w_{n,j,p}^2}} \tag{3.15}$$

3.7.1.3 Resource Similarity (ReSim) The ReSim measure extends Linked Data Semantic Distance (LDSD) to satisfy three fundamental word similarity axioms that are violated, including "equal self-similarity", "symmetry" or "minimality". According to Piao e Breslin (2016), the following axioms must be resolved:

- Equal self-similarity: $sim(A, A) = sim(B, B)$, for all stimuli A and B;
- Symmetry: $sim(A, B) = sim(B, A)$, for all stimuli A and B;
- Minimality: $sim(A, A) > sim(A, B)$, for all stimuli A $\neq$ B.

The similarity measure Resource Similarity (ReSim) aims to calculate the similarity of two resources in DBpedia, considering the similarity of the properties of these resources and satisfying the fundamental axioms pointed out above.

It is important to point out that the similarity measures presented so far and most of the other measures in the literature do not consider the information carried out by RDF literals. Along the remainder of this work, we refer as *pure linked-based method* to any measure that discards RDF literals and calculates the similarity of resources relying solely on the links (direct or indirect) that interconnect them.

Another branch of this work is to investigate ways to automatically select and rank the items' features to provide better recommendations to the user. We will discuss next some ways of ranking features regarding a given RS context and/or selecting the best combination of properties that fits a particular RS.

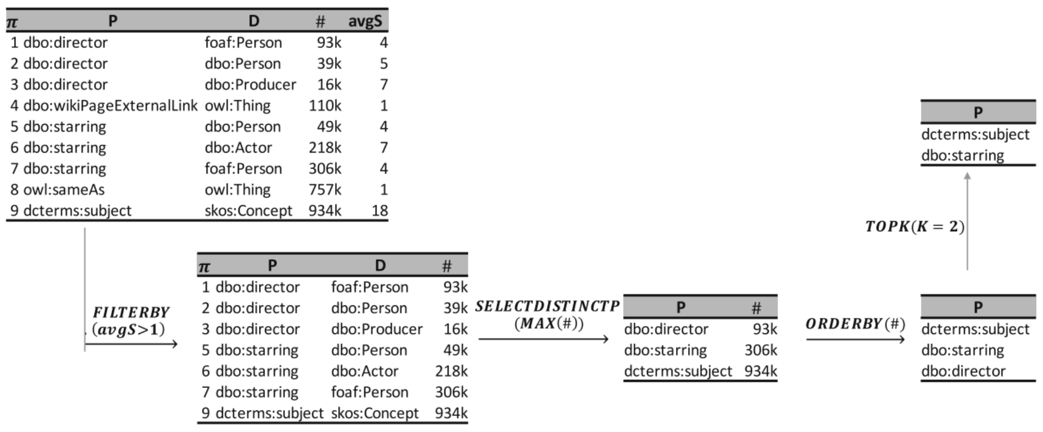
3.8 FEATURE SELECTION AND FEATURE RANKING

In machine learning, feature selection has been used in research areas for which datasets with hundreds or thousands of variables are available. Studies about this issue focus mainly on two branches of research: i) constructing and selecting subsets of useful features to build a good predictor, and; ii) finding and ranking all potentially relevant variables in a context (GUYON; ELISSEEFF, 2003). Both Feature Selection (FS) and Feature Ranking (FR) can be used as a filter method, i.e., a preprocessing step, independent of the choice of the predictor (GUYON; ELISSEEFF, 2003). This preprocessing work helps the system to lead with the high sparsity characteristic of the matrix *User* $\times$ *Item*.

In a LOD-based system, one common way of performing an FS task is through hand-crafted work, i.e., manual selection of the most relevant LOD-based features, according to simple heuristics or the domain knowledge (MUSTO et al., 2016). In a LOD-Based Recommender System, properties can be considered as features of a given node of the knowledge graph. Even though most recommender models consider links as having equal importance to compute similarity, FS and FR have been applied to increase the accuracy of RSs. For example, Musto et al. (MUSTO et al., 2016) assess the impact of several feature selection techniques on recommendations accuracy, such as Principal Component Analysis (PCA), Information Gain Ratio (GR), PageRank (PR), and Support Vector Machines (SVM).

In a different approach, Noia et al. (2018) shows how LOD-based summarization can drive FS and FR tasks by comparing an automated feature selection method based on ontology data summaries with more classical ones, like manual selection. Summarization is an FS method based on a group of descriptors from the graph model. It automatically extracts the top-k properties deemed more important to evaluate the similarity between instances of a given class on top of data summaries built with the help of an ontology. The method uses frequency and cardinality descriptors computed over schema patterns such as ⟨dbo:Film, dbo:starring, dbo:Actor⟩ extracted from the data (NOIA et al., 2018).

The FS task is performed by Noia et al. (2018) though the ABSTAT framework, which provides two statistics: the pattern frequency and the cardinality descriptors for feature selection. The process exemplified in Figure 3.5 shows a subset of Π with class dbo:Film as



π	P	D	#	avgS
1	dbo:director	foaf:Person	93k	4
2	dbo:director	dbo:Person	39k	5
3	dbo:director	dbo:Producer	16k	7
4	dbo:wikiPageExternalLink	owl:Thing	110k	1
5	dbo:starring	dbo:Person	49k	4
6	dbo:starring	dbo:Actor	218k	7
7	dbo:starring	foaf:Person	306k	4
8	owl:sameAs	owl:Thing	757k	1
9	dcterms:subject	skos:Concept	934k	18

π	P	D	#
1	dbo:director	foaf:Person	93k
2	dbo:director	dbo:Person	39k
3	dbo:director	dbo:Producer	16k
5	dbo:starring	dbo:Person	49k
6	dbo:starring	dbo:Actor	218k
7	dbo:starring	foaf:Person	306k
9	dcterms:subject	skos:Concept	934k

P	#
dbo:director	93k
dbo:starring	306k
dcterms:subject	934k

P
dcterms:subject
dbo:starring
dbo:director

P
dcterms:subject
dbo:starring

Figure 3.5: Feature selection model with ABSTAT with source type dbo:Film. Adapted from Noia et al. (2018).

the source type. The first step of this approach (FILTERBY) filters out properties based on the local cardinality descriptors. Mre specifically, it filters only properties for which the average number of distinct subjects associated with unique objects is more than one (avgS > 1), i.e, it consider only those properties connecting one target type with many source types. In the example, patterns $\pi 4$ and $\pi 8$ with dbo:wikiPageExternalLink and owl:sameAs property, respectively, are removed because there exists, on average only one subject of type dbo:Film associated with a distinct object. The second step of the process (SELECTDISTINCTP) selects all properties of the patterns in Π by applying the maximum of the pattern frequency (# in the figure). Then, the properties are ranked (ORDERBY) in descending order on pattern frequency, and then k properties (TOPK) are selected (with $k = 2$ in this example).

In the present work, we perform an FS task in a personalized way by adapting the work of Noia et al. (2018) to the proposed personalization strategies. We select the most domain-relevant features (links) from the LOD graph using an implementation of the ABSTAT framework. After that, we rank the filtered links by adding weights to each link according to the user model (see Chapter 5 for further details). Differently of Noia et al. (2018) that use the Jaccard index as the similarity measure, we combine the FS task with personalized versions of Linked Data Semantic Distance (LDSD) and Resource Similarity (ReSim). In other words, we aim to prove that ranking features according to their relevance to the user, instead of considering just the semantics of domain, increases the accuracy of the RS as we show in Chapter 6.

3.9 SUMMARY

The recommendation algorithms approach different ways of making recommendations. This chapter presented a formal definition of RSs in Section 3.1. Then, user feedback techniques were addressed to collect information regarding their interest in Section 3.3

and the main recommendation algorithms discussed in Section 3.4. After that, the tasks that a given RS performs to improve the user experience were described in Section 3.2 and, in Section 3.5, we discussed some problems that we need to deal with when developing an RS.

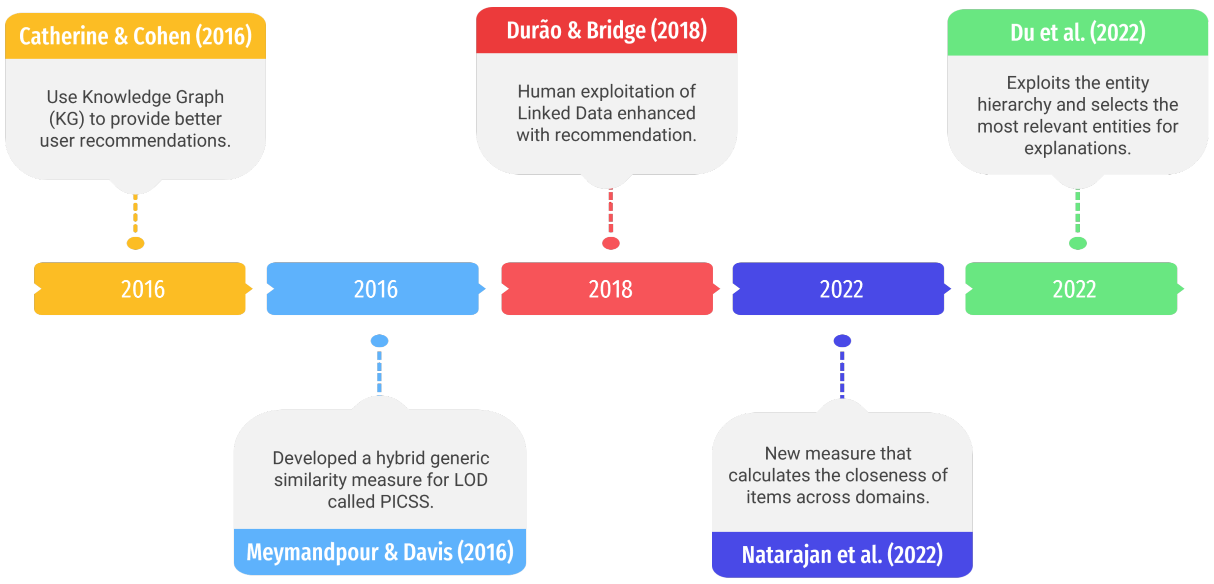
In addition, in Section 3.6, we presented metrics for RS accuracy evaluation. In Section 3.7, we discussed LOD-based Recommender Systems by putting together concepts that were previously discussed in Chapter 2, with the RS concepts that have been shown in this chapter. Finally, we presented LOD-based techniques for RS as semantic similarity measures and feature selection in Section 3.8.

The main idea of this chapter was to present important concepts for the understanding of RS techniques used in the development of this work. Next, in Chapter 5 we will introduce how we used the concepts discussed in this chapter to personalize the user model and thus recommend items better suited to the user's taste.

Chapter 4

RELATED WORK

This research aims to improve recommendations in LOD-based systems through user model personalization techniques. This chapter lists works in the two research areas related to this thesis: i) LOD-based Similarity Measures; and ii) Feature Selection. Figures 4.1 and 4.2 help us to understand the timeline in which the main works were developed. This visual organization shows that the development of research in the two areas has had a similar temporality, which can be explained by the fact that feature selection and recommender systems are connected themes. The following sections provide more details about the related work.



Catherine & Cohen (2016)
Use Knowledge Graph (KG) to provide better user recommendations.
Durão & Bridge (2018)
Human exploitation of Linked Data enhanced with recommendation.
Du et al. (2022)
Exploits the entity hierarchy and selects the most relevant entities for explanations.
2016
2016
2018
2022
2022
Developed a hybrid generic similarity measure for LOD called PICSS.
Meymandpour & Davis (2016)
New measure that calculates the closeness of items across domains.
Natarajan et al. (2022)

Figure 4.1: Timeline of work on LOD-based Similarity Measures.

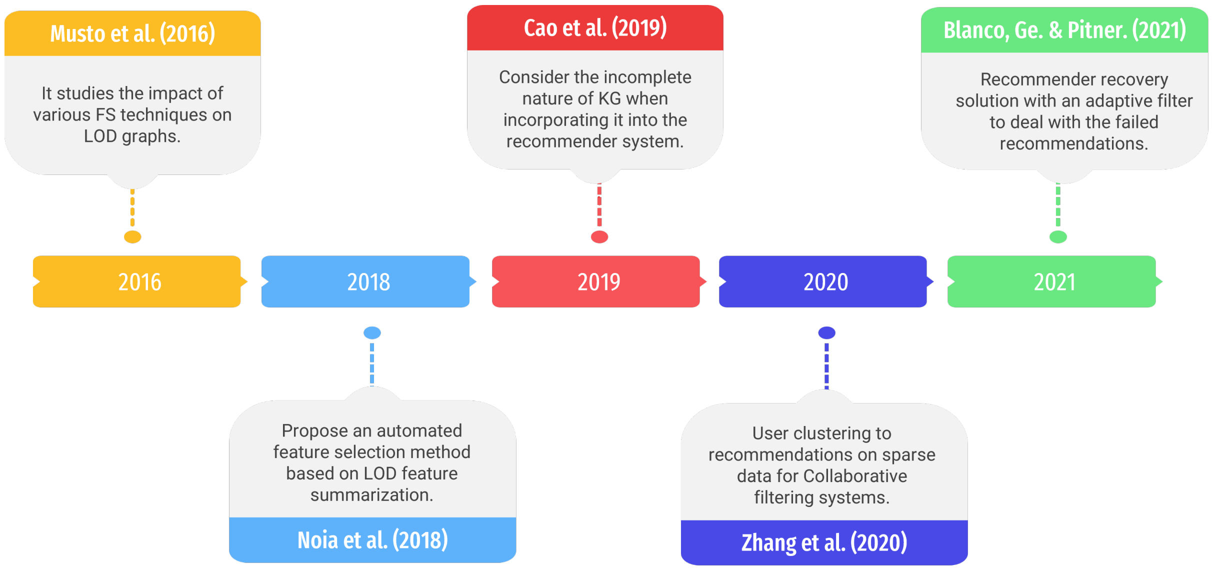

Musto et al. (2016)
It studies the impact of various FS techniques on LOD graphs.
Cao et al. (2019)
Consider the incomplete nature of KG when incorporating it into the recommender system.
Blanco, Ge. & Pitner. (2021)
Recommender recovery solution with an adaptive filter to deal with the failed recommendations.
2016
2018
2019
2020
2021
Propose an automated feature selection method based on LOD feature summarization.
Noia et al. (2018)
User clustering to recommendations on sparse data for Collaborative filtering systems.
Zhang et al. (2020)

Figure 4.2: Timeline of work on Feature Selection.

4.1 LOD-BASED SIMILARITY MEASURES

This work addresses the problem of building a recommender system over a LOD dataset where resources' properties are considered items' features. Several related works draw on a semantic definition of RS, such as Catherine & Cohen (2016), which asset a semantics-aware application where a Knowledge Graph (KG) is used to provide better user recommendations. In most works following this line of research, the authors enrich benchmark databases with semantics given from the interconnections between KG nodes.

Meymandpour & Davis (2016) propose a generalized information content-based approach with systematic semantic similarity assessment between entities. They extensively reviewed existing semantic similarity measures and developed a hybrid method made of feature-based and statistical approaches. Musto et al. (2016) studies the impact of using knowledge coming from LOD on the overall performance of a graph-based recommendation algorithm. Following the line of Meymandpour & Davis (2016), they enrich a graph environment with semantics and test several feature selection techniques against two datasets. Chhatwal & Deepak (2022) incorporate semantic frame matching and entities enriched by the generation of Resource Description Framework and background knowledge from the Linked Open Data cloud.

Natarajan et al. (2022) propose a new measure that calculates the closeness of items across domains. The authors calculate the semantic relatedness instead of similarity because the cross-domain item attributes of semantic resources are diverse. The authors claim that the proposed model provides relevant, personalized recommendations for the target new user with the user preferences gained from the source domain and by exploiting item semantic relatedness. Unlike our work, they focus on solving a cold start problem by looking into a general similarity model considering various entities simultaneously. Our work, like others, is more turned to investigating one-to-one relationships.

In Singh, Sahu & Sharma (2018), user-based collaborative filtering generates recommendations using items and user preferences based on splitting criteria for movie recommendation applications. Based on contextual values, every item and user is split into two virtual items and two virtual users. The recommender technique is then applied to the new dataset of split items and users.

Du et al. (2022) reaffirm that Semantic web resources such as Linked Open Data (LOD) and KG are useful tools for enhancing posthoc recommendation explanations as they are capable of representing facts and domain knowledge in a formal, machine-readable way. The authors claim that existing LOD-based explanation approaches rely mainly on overlapping features (i.e., the KG's entities) between user-liked and recommended items. However, the main drawback of existing methods is that neither the KG's hierarchy nor the hierarchical relationships among entities are considered carefully, which may lose the inference power of knowledge graphs and lead to irrelevant or redundant explanations. To address these issues, the authors propose a generic method that efficiently exploits the entire entity hierarchy and selects the most relevant entities for explanations. Our work does not address the explanation issue but recommendations. However, the concern regarding exploiting hierarchy is relevant and sounds like an interesting approach to enhance our similarity models.

In an effort to improve the human exploitation of this data, Durão & Bridge (2018) propose a Linked Data browser that is enhanced with recommendation functionality. Based on a user's profile, also represented as Linked Data, the authors propose a technique called LDRec that chooses in a personalised way which resources within a certain neighbourhood in a Linked Data graph to recommend to the user. An Iterative Classification Algorithm inspires the novel recommendation technique. This particular work could benefit from our similarity models as they use DBpedia, and they must experience the concerns raised in this thesis.

Yi, Huang & Qin (2018) approach combines the tasks of rating prediction from a rating-based system with a review-based RS. In this manner, they made a user-item rating relation from latent feature representations of the user and item. They extracted the user-item review relation from the user's review content to the item. Then, to fuse these two relations, they proposed extractors represented by auto-encoders based on adversarial learning. The difference between these two relations is minimized according to the more the encoders learn. It is a very interesting approach that can personalize the model but is not fitted to LOD-based environments.

The main difference between the related work cited above and our approach is that none leverage past interactions to build the user model by personalizing the links. We start from the idea that features appearing on positively ranked items have more importance to the similarity engine. This leads to a list of the recommended items more accurate to the users' taste. Some studies have been conducted in this line of user-based personalization. However, most approaches do not encompass Linked Open Data (LOD) systems.

4.2 FEATURE SELECTION

This work addresses the problem of building a recommender system over a LOD dataset, where resources' properties are considered as items' features. This type of system suffers from the matrix sparsity problem, which refers to the issue of having a large amount of data with very few ratings or interactions. This can make it difficult for the system to make accurate recommendations because it may not have enough information about a user's preferences. Meymandpour & Davis (2016) do a consistent literature review of semantic similarity measurements. The classical feature-based similarity measures inspired our search for methods to lead to the sparsity problem through a feature selection approach.

Musto et al. (2016) reduced the matrix dimension by testing several feature selection techniques against two datasets. They investigate whether the integration of LOD-based features improves the algorithm's effectiveness and to what extent the choice of different feature selection techniques influences its performance in terms of accuracy and diversity. Furthermore, Noia et al. (2018) show how LOD-based summarization can drive feature selection tasks by proposing a fully automated method based on semantics provided by the ontology. They evaluate the accuracy of each strategy used and compare it with classic methods such as manual selection and statistical distribution methods. Finally, they aggregate diversity to the recommender system by exploiting the top-k selected features.

In the inverse direction, Van Rossum & Frasincar (2019) investigate the incorporation of graph-based features into path-based similarities. They proposed two normalization procedures that adjust user-item path counts by the degree of centrality of the nodes connecting them. It is based on the idea that a user liking one movie tells us more about the popularity of this movie than the particular user. Our approach has similar premises. However, the methodology diverges as we exploit the link information from each user's perspective.

Some studies cover the field of user-based personalization. However, most of the approaches do not embrace LOD-based systems. In Singh, Sahu & Sharma (2018), a user-based collaborative filtering system generates recommendations by utilizing items and user preferences based on splitting criteria for movie recommendation applications. Considering contextual values, every item and user is split into two virtual items and two virtual users. The recommender technique is then applied to the new dataset of split items and users. A new approach is proposed by Yi, Huang & Qin (2018) by combining the tasks of rating prediction from a rating-based system with a review-based RS. In this manner, they made a user-item rating relation from latent feature representations and fused it to the user-item review relation extracted from users' reviews.

The latest research addresses new approaches to improve the user experience in RS, as in Blanco, Ge. & Pitner. (2021). They propose a recommender recovery solution with an adaptive filter to deal with the failed recommendations. After a recommendation failure, this solution filters out all items similar to the one disliked by the user. The objective is to keep the user engaged and allow the recommender system to become a long-term application. Like in Yi, Huang & Qin (2018), this approach is not adapted to LOD-based systems.

Gan et al. (2021) propose an EM-model that alternates between a general item diversity learning and knowledge graph embedding learning for user and item representation, which helps to achieve better results in comparison to the state-of-art baselines on datasets MovieLens and Anime. Although this work takes advantage of auxiliary information and historical interactions between user and item from knowledge graphs, it does not cover a personalization method. Another approach is exploited by Zhang et al. (2020), which applies User clustering to recommendations on sparse data for Collaborative filtering systems. Unlike other works, they use user clustering to reconstruct the user-item bipartite network to improve the network density. The recommendation made on this dense network thus can achieve much higher accuracy than on the original sparse network (ZHANG et al., 2020). Cao et al. (2019) proposed a research based on the idea that a KG commonly has missing facts, relations, and entities. Thus, they argue that it is crucial to consider the incomplete nature of KG when incorporating it into the recommender system.

Among the related works, this is the one that most approach our research because they explore the KG trying to understand why a user likes an item. They provide an example that if a user has watched several movies directed by (relation) the same person (entity), it is possible to infer that the director relation plays a critical role when the user makes the decision, thus helping to understand the user's preference at a finer granularity (CAO et al., 2019).

Finally, the main difference between the related works cited above and our approach is that neither leverages the user model (KG) knowledge to make personalized recommendations. We started our research from this idea and developed a weighting algorithm that ranks properties by importance from the user's perspective. Thus, before the ranking step, we also address the concept of KG completion, although we use the rationale of direct and indirect links coming from Passant (2010). We generate the missing direct links by counting the indirect links between the items the user has interacted with, as demonstrated in Chapter 5.

4.3 SUMMARY

In this chapter, we discussed some related research works that cover the scope of LOD-based RS in a context of personalizing the recommendation models. For each group of works, a comparison has been made with the present work, highlighting the differences in focus and strategies.

Chapter
5

EXPLOITING LOD-BASED SIMILARITY PERSONALIZATION STRATEGIES FOR RECOMMENDER SYSTEMS

This chapter is dedicated to explaining how our proposed solution works. It begins with an overview of the generic solution and follows with background information supporting the proposal branches, encompassing some built notation and formalism. Then, in the following subsections, we present the details about each step of our solution, making correlations with the Specific Goals proposed in Chapter 1.

5.1 SOLUTION OVERVIEW

This work explores personalization strategies for calculating Linked Data Semantic Similarity in LOD-based Recommender Systems. We develop automatized approaches to personalize the user model, such as assigning weights to the links in the LOD graph, selecting the best features, and exploiting the similarity of literal properties and links. This research aims to improve the precision of recommendations in LOD-based environments of diverse domains.

That being said, Figure 5.1 shows the solution overview, which illustrates the main branches — or steps — of our proposal and how these steps work together to achieve personalized user recommendations. In the following list, we summarize these steps.

1. **Feature Selection:** The first step plays the role of selecting a subset of the most predictive features to increase the performance of the recommendations. Baseline datasets are enriched with DBpedia resources, and an RDF-based Feature Selection algorithm is performed. The output from this phase of the architecture is a Filtered LOD Database. However, most existing feature selection methods select only a fixed subset of features according to the system's domain. This is why we perform the personalization step after the FS step.

2. **Graph Personalization:** This is a preprocessing step in which we automatically weigh the features by analyzing the user's previous preferences, independent of domain. Thus, this step generates a database of ranked features by combining the user model and the input LOD database, where features that are influenced better by the user's choices receive a heavier weight.

3. **Recommender Model:** It goes through how the weighted user model is applied with similarity measures, turning them into personalized semantic similarity measures for recommending items to the user. Our solution creates a personalized recommender system that supports similarity measures based on links and literals. The outputs from this last step are the recommendations themselves.

   (a) **Similarity of Links:** Refers to the similarity measures developed to calculate how similar items in an RDF graph are based on the distance between their links. These measures consider only links between two resources, i.e., object property links. For example, the dbo:director property links a movie to the person who directed it.

   (b) **Similarity of Literals:** This Is a set of methods that leverage the semantics contained in the literal properties of the RDF resource. In other words, it calculates the similarity between two LOD resources based on the semantics of the shared datatype properties. This step is applied when the user desires to leverage the semantics of the text in an RDF graph. For example, the dbo:abstract property from a movie.

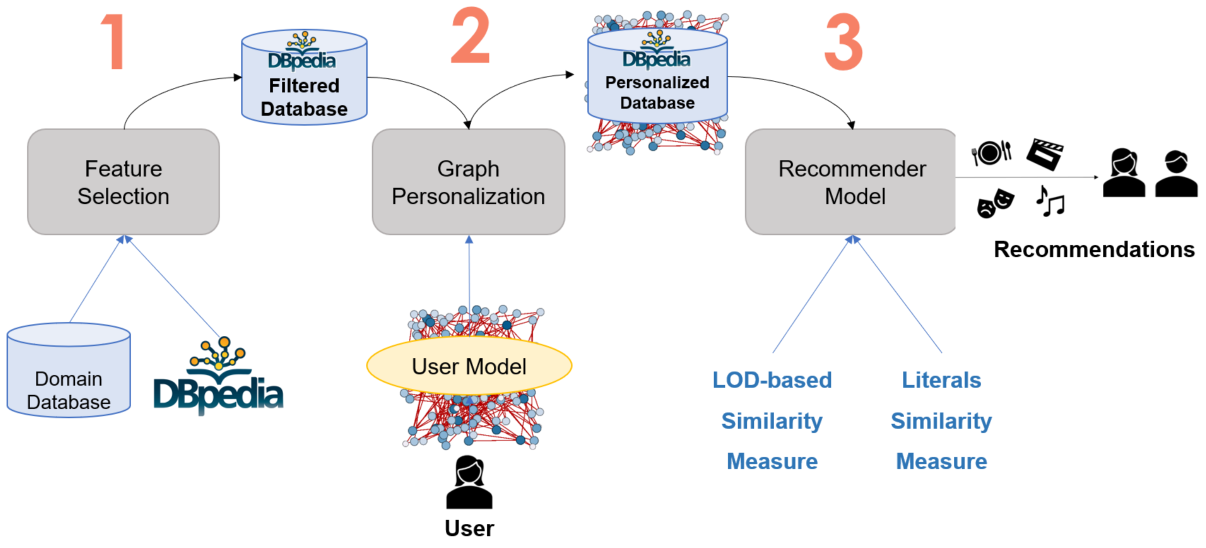


Figure 5.1: The solution overview.

In the experimental phase, we apply the adapted Summarization FS method (NOIA et al., 2018) in step 1; the authorial weighting personalization method (SILVA; DURãO; CAPRETZ, 2019) in step 2; and the baseline methods Linked Data Semantic Distance

(LDSD) (PASSANT, 2010) and Resource Similarity (ReSim) (PIAO; ARA; BRESLIN, 2016) in step 3. However, the solution proposed in this work is generic enough to allow for any other measure or technique that suits each stage of the model.

5.2 BACKGROUND AND NOTATIONS

This section is dedicated to clarifying the mathematical formalism created to clearly and objectively describe this proposed research solution. This formalism, hereafter referred to as background notations, will be introduced as the solution steps are presented. To simplify understanding, Figure 5.2 will be used as an illustration for most of the usage examples shown in the following subsections.

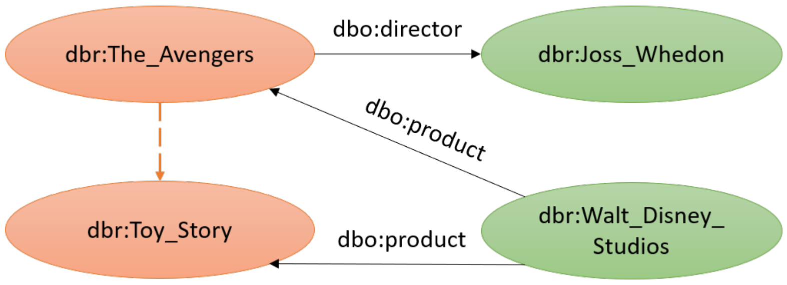


Figure 5.2: RDF graph example.

5.2.1 The Linked Data Graph

We developed a mathematical notation, similar to Passant (PASSANT, 2010), to represent any dataset that follows the Linked Data principles: A LOD graph is a directed graph $G = (R, P, T)$, in which $R = \{r_1, r_2, ..., r_n\}$ is a set of resources identified by their URI; $P = \{p_1, p_2, ..., p_n\}$ is a set of properties identified by their URI; and $T = \{t_1, t_2, ..., t_n\}$ is a set of triples — instances of properties linking pairs of resources. Thus, as an RDF triple is a statement in the format ⟨subject, predicate, object⟩, then $t_i = \langle r_a, p_j, r_b \rangle \in T$ means that there is an instance of a property $p_j \in P$ linking the subject $r_a \in R$ to the object $r_b \in R$, like in ⟨dbr:The_Avengers, dbo:director, dbr:Joss_Whedon⟩, back to Figure 1.2.

5.2.2 The User Model

A user model is how a recommender system represents users' preferences about items, as discussed in Chapter 3 Section 3.3. In this work, we model it as part of the same Linked Data graph. Therefore, we assume a set of users $U = \{u_1, u_2, ..., u_n\}$ in which each u_k is also itself a resource in the LOD graph, i.e., $U \subset R$. We also assume the existence of properties that represent how much the users like items on a scale from 1 to 5, wrapped in the set $P' = \{p'_1, p'_2, p'_3, p'_4, p'_5\} = \{hasRatedAs1, hasRatedAs2, hasRatedAs3, hasRatedAs4, hasRatedAs5\}$, with $P' \subset P$. Having said this, we bring to light the set T' formed by triples in the format $t'_i = \langle u_k, p'_l, r_a \rangle \in T'$. For example, a triple $t'_1 = \langle u_1, hasRatedAs5, r_1 \rangle$ means that user $u_1 \in U$ has rated the item $r_1 \in R$ as 5 stars.

Thus, we can define the total reach of our modeling as $G \cup G'$. In other words, the union between graph $G = (R, P, T)$ and the user model graph $G' = (R, P', T')$.

5.2.3 The Rating Scale

Ratings in recommender environments are positive or negative explicit feedback given by the users according to how much they like or dislike the items (RICCI; ROKACH; SHAPIRA, 2015). In our model, we use a 1 to 5 stars Likert scale. However, at the weight calculation step, we convert the stars to an integer scale that emphasizes lower feedback by assigning negative values to them. This scale functions as follows: 5 stars mean that the user likes the movie, so we count this movie times 3; rating 4 means that the user likes the movie, so we count it times 2; rating 3 means that the user does not like the movie nor dislike it, so this movie counts times 1; rating 2 means that the user dislikes the movie, so we count this movie times -2; and, finally, rating 1 means that the user really dislikes the movie and then we count it times -3. Thereby, the lowest star ratings — 2 and 1 — are considered negative user feedback. Table 5.1 summarizes the above rating system.

Table 5.1: Adopted rating system summarization.

5-Star rating	Property name	Integer value
★	hasRatedAs1	-3
★★	hasRatedAs2	-2
★★★	hasRatedAs3	1
★★★★	hasRatedAs4	2
★★★★★	hasRatedAs5	3

5.2.4 The data returned by ABSTAT

To perform the Feature Selection (FS) preprocessing step, which will be discussed in 5.3, we use the ABSTAT API[1] from Noia et al. (2018). These tools process a linked database and extract relevant summarized information, with statistics about each ontology element used in the dataset. The summarized data consist of an aggregate of triples, their frequencies, and local and global cardinality descriptors. Among the data returned by ABSTAT Application Programming Interface (API), we use:

- T - Summarized data set, including the triples (s, p, o), the frequencies (f), and the averages of distinct subjects associated with a single object $(avgS)$. Table 5.2 demonstrates an example of the elements contained in T;
- (s, p, o) - RDF triple, where s, p, and o are considered subject, property, and object, respectively. Each element of the triple is represented by a Uniform Resource

[1]http://abstat.disco.unimib.it/

Table 5.2: Example of elements that belong to the data set returned by ABSTAT.

s	**p**	**o**	**F**	**avgS**
dbo:Film	dbo:wikiPageWikiLink	foaf:Person	12k	10
dbo:Film	purl:subject	skos:Concept	99k	7

Identifier (URI). For example, in the triple (dbo:Film, dbo:wikiPageWikiLink, foaf:Person), there is a subject of type dbo:Film connected to an object of type foaf:Person through the dbo:WikiPageWikiLink property;

- *avgS* - Cardinality describing the average of distinct subjects associated with a single object in the dataset extension. That is, if for a triple of type (dbo:Film, dbo:wikiPageWikiLink, foaf:Person) we have that avgS = 10, then we have the average of 10 distinct subjects of type dbo:Film connected to a single object of type foaf:Person via the dbo:WikiPageWikiLink property.

- f - Represents the frequency pattern of each triple, i.e., reports the number of instances times (instances) a pattern (s, p, o) was found in the database extension. For example, if $f = 12k$ for the triple (dbo:Film, dbo:wikiPageWikiLink, foaf:Person), there are 12 thousand occurrences of that triple pattern in the database.

Besides the values returned by ABSTAT, we also use the variable k to indicate the number of properties that should be selected in the pre-processing stage.

5.3 STEP 1: FEATURE SELECTION

This section explains our approach to achieve specific goal *SG1: Propose a feature selection approach to filter relevant properties according to the domain* (as defined in Chapter 1). We developed the first step of the architecture model (as seen in Figure 5.1) to be a preprocessing step for Feature Selection, which will feed into the next step (the personalization step).

Noia et al. (2018) proposed the filtering algorithms used in this work to automatically extract the k properties considered most important for evaluating the similarity between instances of a given class. We adapted the agorithms of the ABSTAT framework to evaluate the similarity between DBpedia instances of type dbo:Film and dbo:MusicalArtist, through three experiments described in Chapter 6.

The proposed feature selection uses two statistics provided by the ABSTAT framework API (NOIA et al., 2018). Among these, we use the value of a cardinality descriptor, representing the averages of distinct subjects associated with a single object in the analyzed database extension and the frequencies of each RDF triple pattern. The proposed preprocessing algorithm is divided into four distinct steps: i) filtering the properties through the averages of distinct subjects associated with a single object; ii) selecting the distinct properties through the highest frequencies; iii) sorting these properties; and iv) selecting the k most relevant properties. The Summarization (Algorithm 1) represents the union of these steps and Figure 5.3 demonstrates the order in which they occur.

Algorithm 1: The Summarization Algorithm

```
  Input: $T, K$
  Output: $topKProperties$
1 Function summarization($T, K$):
2     $filteredT \longleftarrow filterBy(T)$;
3     $maxDistinctP \longleftarrow selectDistinct(filteredT)$;
4     $orderedProperties \longleftarrow orderBy(maxDistinctP)$;
5     $topKProperties \longleftarrow selectTopK(orderedProperties, K)$;
6     return $topKProperties$;
7 End Function
```

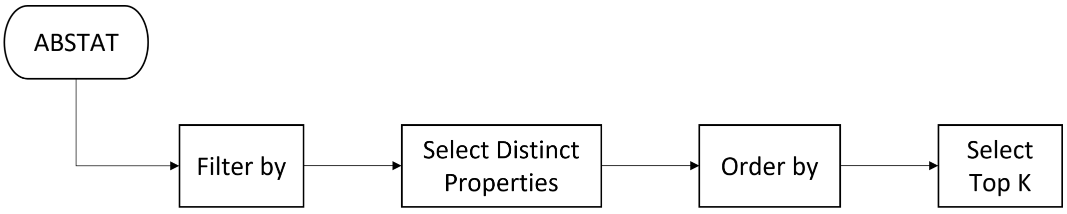


Figure 5.3: The ABSTAT adapted algorithms flowchart.

The aforementioned steps have been incorporated into this project in the feature selection preprocessing engine, which aims to reduce the feature matrix dimension and consequently improve the system's performance. Each algorithm in this approach will be discussed in the following subsections.

5.3.1 Filtering ($filterBy$)

The Filtering step (Algorithm 2), represented by the $filterBy$ method, selects, from the dataset extent, the RDF triples in which the average number of distinct subjects associated with a single object is greater than one. Thus, the $filterBy$ method receives the set T as a parameter, in which each element of T represented by t contains a triple (s, p, o), a frequency f, and an average $avgS$. For each t, the method checks whether $avgS$ is greater than one. If so, t is added to the list $filteredT$ that will be returned by the method. The notations used here have been discussed in more detail in 5.2.4. The filtering results are passed as parameter to the next step: Select Distinct Properties.

5.3.2 Selecting Distinct Properties ($selectDistinct$)

The $selectDistinct$ method (Algorithm 3) is used in this step to remove duplicated properties, keeping only those that appear once in the set of triples or those with the highest frequency. This method receives the filtered dataset with properties and their respective frequencies and checks whether each property is unique. If the property is not unique, the one with the highest associated frequency value is selected. The function returns a list of distinct properties and their frequencies, which are used in the sorting process (refer

Algorithm 2: The Filtering Algorithm

```
Input: $T$
Output: $filteredT$
Function filterBy($T$):
    $filteredT \longleftarrow []$;
    foreach $t \in T$ do
        if $t.avgS() > 1$ then
            $filteredT \longleftarrow filteredT + t$;
        end
    end
    return $filteredT$;
End Function
```

to Algorithm 4).

Algorithm 3: The Select Distinct Properties Algorithm

```
Input: $filteredT$
Output: $maxDistinctP$
Function selectDistinct($filteredT$):
    $maxDistinctP \longleftarrow []$;
    foreach $t \in filteredT$ do
        if $t.p \in maxDistinctP$ then
            $pElement \longleftarrow maxDistinctP[t.p]$;
            if $t.p.getF() > pElement.getF()$ then
                $maxDistinctP[t.p] \longleftarrow t.p$;
            end
        else
            $maxDistinctP \longleftarrow maxDistinctP + t.p$;
        end
    end
    return $maxDistinctP$;
End Function
```

5.3.3 Sorting (*orderBy*)

This step is responsible for sorting the properties according to their frequencies in descending order. This way, the properties with the highest frequency values will be at the top of the list and prioritized in the k top properties selection step, as in Algorithm 4. The *orderBy* method is responsible for performing this ordering. The data from this method will be used in the top k properties selection step.

Algorithm 4: The Sorting Properties Algorithm

```
  Input: $maxDistinctP$
  Output: $orderedProperties$
1 Function orderBy($maxDistinctP$):
2     $orderedProperties \longleftarrow sort(maxDistinctP)$;
3     return $orderedProperties$;
4 End Function
```

5.3.4 Selecting the top k properties Algorithm ($selectTopK$)

In this step (Algorithm 5), the *selectTopK* method receives as parameters the set of properties ordered previously and the number of properties to be selected for later use in calculating the similarity metrics. The first k items in the list of sorted properties are then returned.

Algorithm 5: The Select Top K Algorithm

```
  Input: $orderedProperties, k$
  Output: $topKProperties$
1 Function selectTopK($orderedProperties, k$):
2     $topKProperties \longleftarrow []$;
3     for $i \longleftarrow 0, i < k$ do
4         $topKProperties \longleftarrow topKProperties + orderedProperties[i]$;
5     end
6     return $topKProperties$;
7 End Function
```

5.3.5 Running Example

To exemplify the recommendation proposed in this research, this section is divided into two subsections, where the first one will demonstrate an example of the preprocessing of features presented in Section 5.3 (using notations and methods described in such section), and the second one will demonstrate an example of the application of this FS preprocessing in the recommendation process.

5.3.5.1 Features Preprocessing To select the best properties, we firts need to query the ABSTAT framework API to retrieve the list of distinct subject properties, objects, frequencies, and averages associated with a single object in the database extension. Consider that a query is made, and the elements of Table 5.3 are returned. Thus, after the API query, the $filterBy$ method will select only elements whose $avgS > 1$, as shown in Table 5.4.

The return of $filterBy$ method will then be used by the *selectDistinct* method to select the distinct properties with the highest frequencies. The result of running this

Table 5.3: Elements returned from the ABSTAT API.

P	**O**	**F**	**avgS**
dbo:wikiPageWikiLink	foaf:Person	12k	10
purl:subject	skos:Concept	99k	7
dbp:wikiPageUsesTemplate	owl:Thing	93k	7
dbo:starring	dul:Agent	28k	3
dbp:title	dbo:Single	2k	3
dbp:music	dbo:MusicalArtist	57k	1
dbo:wikiPageWikiLink	owl:Thing	55k	5
purl:subject	owl:Thing	1k	4
dbo:starring	dbo:Actor	12k	2
dbp:footer	rdf:langString	266	1
dbp:distributor	dbo:Company	44k	1

Table 5.4: Data returned from the $filterBy$ method.

P	**O**	**F**	**avgS**
dbo:wikiPageWikiLink	foaf:Person	12k	10
purl:subject	skos:Concept	99k	7
dbp:wikiPageUsesTemplate	owl:Thing	93k	7
dbo:starring	dul:Agent	28k	3
dbp:title	dbo:Single	2k	3
dbo:wikiPageWikiLink	owl:Thing	55k	5
purl:subject	owl:Thing	1k	4
dbo:starring	dbo:Actor	12k	2

method can be seen in Table 5.5.

Table 5.5: Data resulting from the *selectDistinct* method.

P	**F**
purl:subject	99k
dbp:wikiPageUsesTemplate	93k
dbo:starring	28k
dbp:title	2k
dbo:wikiPageWikiLink	55k

After the execution of *selectDistinct*, the *orderBy* method will sort the properties according to their frequencies, and this sorted list will be used by the *selectTopK* method. This method will return the k properties from the top of the list. The return of these last two methods can be seen in Table 5.6, considering $k = 4$.

Table 5.6: Data returned from *orderBy* e *selectTopK* methods, considering $k = 4$.

P
purl:subject
dbp:wikiPageUsesTemplate
dbo:wikiPageWikiLink
dbo:starring

5.3.5.2 Features Preprocessing application As a example of usage, it is supposed that a given user has rated a movie with 5 stars and its properties are represented in Table 5.7 as follows.

Table 5.7: Model of a movie rated by a given user.

Movie 1
dbo:wikiPageWikiLink
dbo:producer
dbo:wikiPageExternalLink
dbo:wikiPageID
dbp:producer
dbo:starring
dbp:title
dbp:wikiPageUsesTemplate
dct:subject
dbp: lengt

Thus, considering the movies in Table 5.8 (Movie 2 and Movie 3), we want to know which one should be recommended to the user. To do this, we will analyze which of them is most similar to Movie 1, which was previously rated by the user. We will use the Feature Selection methods to automatically pre-select the best properties to calculate the similarity of the items involved. Thus, using the preprocessing example, presented in 5.3.5.1, we reach the configuration shown in Table 5.9, which represents the intersection of the four pre-selected properties of Table 5.6, with Tables 5.7 and 5.8.

Table 5.8: Data model for the recommendation.

Movie 2	**Movie 3**
dbo:wikiPageWikiLink	dbo:wikiPageWikiLink
dbo:producer	dbo:producer
rdfs:comment	dbo:wikiPageExternalLink
dbo:wikiPageID	dbp:screenplay
dbp:title	dbo:starring
dbo:distributor	dbo:wikiPageLength
dbo:musicComposer	dbp:country
dct:subject	dbp:footer
dbo:editing	dbp:music
dbo:starring	dbp:productionCompanies

Table 5.9: Properties that will be evaluated on each film.

Movie 1	**Movie 2**	**Movie 3**
dbo:wikiPageWikiLink	dbo:wikiPageWikiLink	dbo:wikiPageWikiLink
dct:subject	dct:subject	-
dbp:wikiPageUsesTemplate	-	-
dbo:starring	dbo:starring	dbo:starring

After that, we have excluded the properties considered irrelevant for calculating the similarity between movies. Now we are able to apply the personalization step, as seen in Section 5.4, in which the weights of the remaining properties are calculated according to the user's preferences. Thus, considering all the movies evaluated by the user, we hypothetically will have the personalization step ordering the four properties of Movie 1 as follows, from the most important to the least important to the user: i) dct:subject; ii) dbo:starring; iii) dbo:WikiPageWikiLink; iv) dbp:WikiPageUsesTemplate.

In the next step, the similarities between Movie 1 and Movie 2 and between Movie 1 and Movie 3 are calculated using LDSD with and without the weights of its properties. It is not possible to assume which film would be recommended to the hypothetical user by the LDSD just by looking at Table 5.9. However, it is very likely that if Movie 1 and Movie 2 have the same value of dct:subject, Movie 2 will be recommended by PLDSD instead of Movie 3 because dct:subject was ranked as the most important property to the user in the personalization step — and is also missing in Movie 3.

Although the example given in this section is hypothetical, at the end of each experiment performed in Chapter 6, the results will be statistically evaluated to verify the impact of applying the proposed strategy in a near-real recommendation system.

5.4 STEP 2: GRAPH PERSONALIZATION

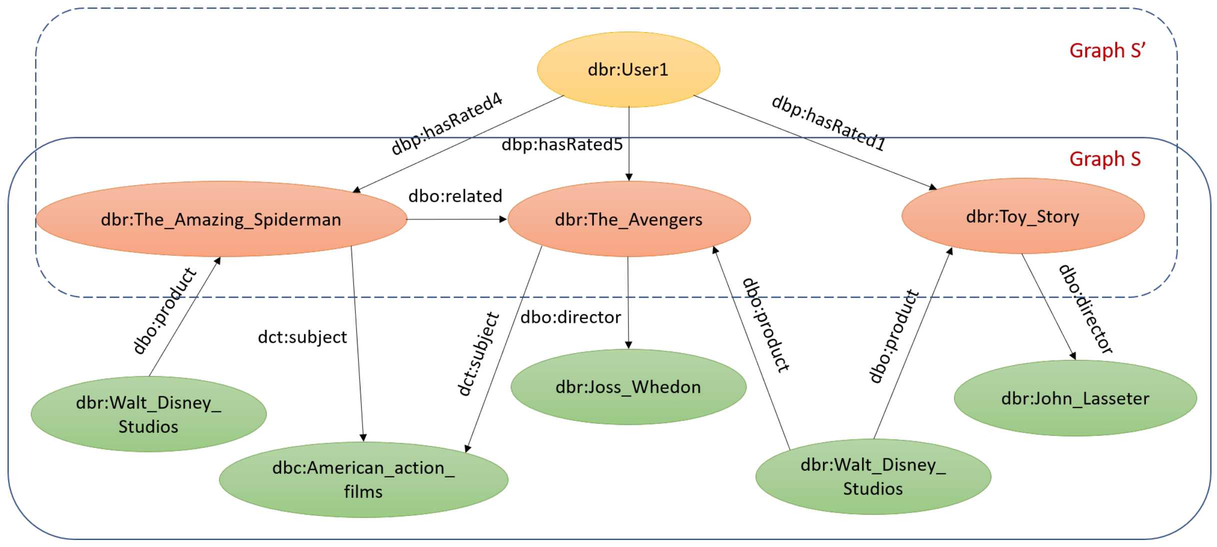


Figure 5.4: Dataset D: a portion of the DBpedia dataset, describing a movie domain.

The second step in the architecture model (as seen in Figure 5.1) is the Graph Personalization step. We developed the method stated in *SG2: Propose a personalization methodology that weighs links in a LOD graph based on the user's past ratings on the items in a recommender system* (Chapter 1). This method analyzes the previous preferences made by a user in a given recommender system, and weights are assigned to the graph's edges/links according to it.

After running the personalization step, the resultant graph accomplishes the specific goal *SG3: Propose a user profile modeling from the personalized graph obtained in the previous step* (see Chapter 1).

5.4.1 Running Example

The following scenario is taken as a running example: a portion of the DBpedia dataset describing a movie domain, called dataset $D \subset (G \cup G')$ — see Figure 5.4.

Figure 5.4 shows the dataset D divided into two subsets. The uppermost set S' is a subgraph of graph G', representing the user model. Thus, graph S' wraps the user, colored in yellow — which means that the `rdf:type` is `dbo:User` — and the main content resources, colored in pink — which means that the `rdf:type` is `dbo:Film`. Hence, only the movies previously rated by the user are elements of S'. The lowermost set S is a subgraph of graph G and shows, besides elements from the user model S', other content resources that are linked to the movies. Those resources are colored in green, which means that they have other `rdf:type` than `dbo:Film`.

Each arrow in the model means one property, or predicate, that links two resources composing one triple in the format ⟨subject, predicate, object⟩. We have then tabulated all the triples from S' on Table 5.10 using the notation $t'_i = \langle u_k, p'_l, r_a \rangle \in T'$. Thus, Table 5.10 summarizes the user model for user u_1. We also added one last column that shows $Rat_{u_k r_a}$: The rating value on r_a given by u_k, which is equal to the property p'_l converted to an integer value using Table 5.1. For instance, triple $t'_1 = \langle u_1, p'_4, r_1 \rangle =$ ⟨dbr:user1, dbp:hasRatedAs4, dbr:The_Amazing_Spider-Man⟩ has a $Rat_{u_k r_a}$ value equal to 2.

Similarly, all the triples from S have been tabulated on Table 5.11 using the notation $t_i = \langle r_a, p_j, r_b \rangle \in T$. Thus, Table 5.11 wraps a collection of triples t_i from graph S, which are all and only the triples that are linked to movies from set S' (the user model for u_1), such as triple $t_1 = \langle r_1, p_1, r_2 \rangle =$ ⟨dbr:The_Amazing_Spider-Man, dbo:related, dbr: The_Avengers⟩. In addition, we remark that one resource can play the subject role as `dbr:The_Avengers` in $t_3 = \langle r_2, p_2, r_4 \rangle =$ ⟨dbr:The_Avengers, dct: subject, dbc:American_-action_films⟩ and, at the same time, can play the object role as in $t_5 = \langle r_6, p_3, r_2 \rangle =$ ⟨dbr:Walt_Disney_Studios, dbo:product, dbr:The_Avengers⟩. For the sake of simplicity, we invert the triples where movies are objects and make them always the subject of statements. Table 5.11 shows those statements already as inverted triples — t_5 is represented as $\langle r_2, p_3, r_6 \rangle$, for instance.

Table 5.10: Collection of triples t'_i from graph S' describing one sample user model.

Triple — t'_i	Subject — u_k	Predicate — p'_l	Object — r_a	$Rat_{u_k r_a}$
t'_1	u_1 — dbr:user1	p'_4 — dbp:hasRatedAs4	r_1 — dbr:The_Amazing_Spider-Man	2
t'_2	u_1 — dbr:user1	p'_5 — dbp:hasRatedAs5	r_2 — dbr:The_Avengers	3
t'_3	u_1 — dbr:user1	p'_1 — dbp:hasRatedAs1	r_3 — dbr:Toy_Story	-3

Table 5.11: Collection of triples t_i from graph S showing one running example on weights calculation.

Triple — t_i	Subject — r_a	Predicate — p_j	Object — r_b	$Freq(r_b)$	$Rat_{u_k r_a}$	$W(p_j, u_k)$
t_1	r_1 — dbr:The_Amazing_Spider-Man	p_1 — dbo:related	r_2 — dbr:The_Avengers	1	2	0.4
t_2	r_1 — dbr:The_Amazing_Spider-Man	p_2 — dct:subject	r_4 — dbc:American_action_films	2	2	1.0
t_3	r_2 — dbr:The_Avengers	p_2 — dct:subject	r_4 — dbc:American_action_films	2	3	1.0
t_4	r_1 — dbr:The_Amazing_Spider-Man	p_3 — dbo:product	r_5 — dbr:Columbia_Pictures	1	2	0.13
t_5	r_2 — dbr:The_Avengers	p_3 — dbo:product	r_6 — dbr:Walt_Disney_Studios	2	3	0.13
t_6	r_3 — dbr:Toy_Story	p_3 — dbo:product	r_6 — dbr:Walt_Disney_Studios	2	-3	0.13
t_7	r_2 — dbr:The_Avengers	p_4 — dbo:director	r_7 — dbr:Joss_Whedon	1	3	0
t_8	r_3 — dbr:Toy_Story	p_4 — dbo:director	r_8 — dbr:John_Lasseter	1	-3	0

We will keep using Figure 5.4 and tables 5.10 and 5.11 to explain how the rating system described before is used to personalize graph D according to one user $u_k \in U$, turning it into a weighted graph. Please refer to Equation 5.1 while we explain the process as follows. We first calculate $Freq(r_b)$ from graph S by iterating over triples $t_i = \langle r_a, p_j, r_b \rangle \in T$, with i varying from 1 to n (see Table 5.11). While iterating, we count how many times each resource appears in the object column in Table 5.11, i.e., how many times it takes the position of r_b in t_i triples. Results are represented in Table 5.11 as $Freq(r_b)$. Having done that, we do an iterated summation over triples $t_i = \langle r_a, p_j, r_b \rangle \in T$, with i varying from 1 to n. In each iteration, we find the $Rat_{u_k r_a}$ value associated with r_a and multiply the object frequency $Freq(r_b)$ by it, as shown in Equation 5.1.

After repeating this process for every $t_i = \langle r_a, p_j, r_b \rangle \in T$, we calculate the mean value for each property p_j. Additionally, if the same property p_j appears linked to more than one object r_b in graph S, values are summed up before calculating the mean. Similarly, if the same property p_j appears for more than one movie r_a in graph S, values are summed up. Finally, we normalize the values, finding $W(p_j, u_k)$, the normalized average weight value for each property p_j given a user u_k (see Equation 5.1). $W(p_j, u_k)$ values for the running example — for user u_1 — are shown in the last column of Table 5.11. Figure 5.5 shows the resulting weighted graph D, including the virtual links created by sharing indirect incoming and outgoing links.

$$W(p_j, u_k) = \frac{1}{1 + \frac{\sum_{t_i} Freq(r_b).Rat_{u_k ra}}{n}} \tag{5.1}$$

where $t_i = \langle r_a, p_j, r_b \rangle \in T$.

This whole graph personalization process is based on the notion that the more a given object is associated with movies that one user liked, the greater property's importance is in calculating the next recommendations for the user. For instance, user u_1 has liked two movies whose subject is American action films (The Amazing Spider-man and The Avengers). Thus, we assume that the property `dct:subject` is an important criterion to user u_1, just as is `dbo:related`, and rank these properties with a greater weight. We will discuss next, how these weights are used to recommend new movies to the user.

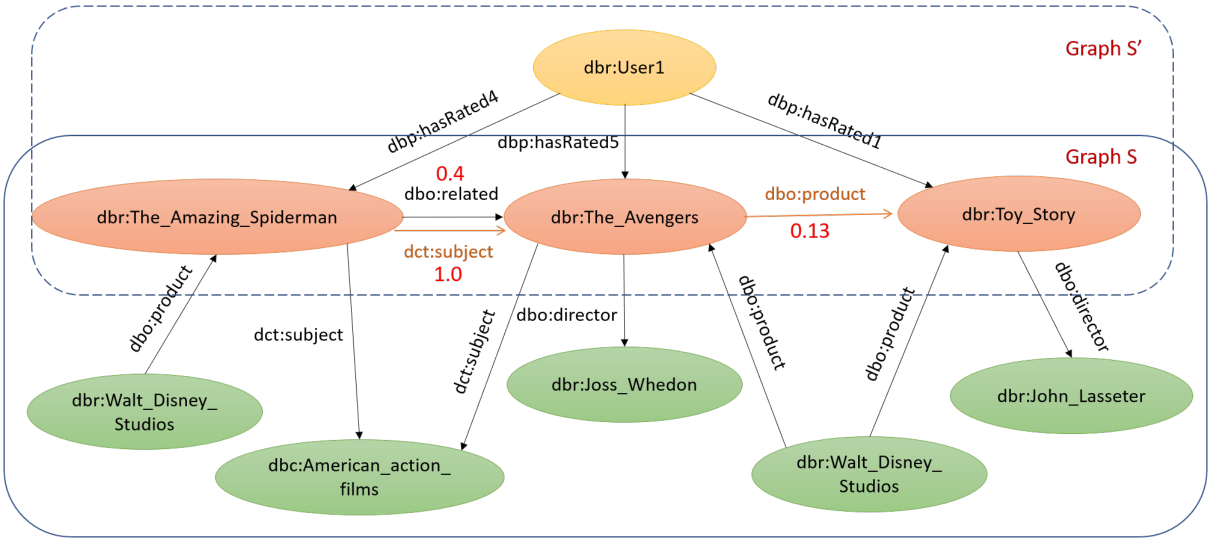


Figure 5.5: Dataset D weighted.

5.5 STEP 3: RECOMMENDATION MODEL

The third step in the architecture model (as seen in Figure 5.1) is the Recommendation Model. The proposed model relies on a similarity measure that combines the personalization methods previously described with one or more LOD-based semantic methods. By assigning weights to the graph, we represent more accurately the user's preferences under a given content-based RS. Also, we leverage the semantic contained in the datatype properties — i.e. those linked to literal objects — of the graph by adding the Literals Similarity (LiSim) step to the algorithm.

All of this means that our approach can be applied to any LOD-enabled CB Recommender System where both domain knowledge and user personalization play an important role. As we use previous information to feed the user model — i.e. ratings given by the user to items in the past —, our approach helps the system to handle the well-known cold-start problem when a new item is added to it. Nevertheless, our proposal still faces the cold-start problem every time a new user is added. However, as we rely on content information at first, this problem is easily overcome by assigning the same weight to all links in the model. Having said that, we will describe next, how the personalization step integrates with two different semantic similarity measures to recommend items to a user.

5.5.1 Personalized Linked Data Semantic Distance (PLDSD)

The classic Linked Data Semantic Distance (LDSD) measure explained in Chapter 3, Subsection 3.7.1, considers every link p_j as having the same weight on calculations. In order to make the recommender system personalized to one user u_k, we have added the function $W(p_j, u_k)$ to Equation 3.14, resulting in Equation 5.2 — the calculus of $W(p_j, u_k)$ has been shown before in Equation 5.1. As $W(p_j, u_k)$ represents the weight to one property p_j given one user u_k, its value is multiplied by every $C(p_j, r_a, r_b)$ function from Equation 3.14, as shown in Equation 5.2.

$$sim_{PLDSD}(r_a, r_b, u_k) = \frac{1}{1 + \sum_j \frac{C_d(p_j,r_a,r_b).W(p_j,u_k)}{1+\log C_d(p_j,r_a,n)} + \sum_j \frac{C_d(p_j,r_b,r_a).W(p_j,u_k)}{1+\log C_d(p_j,r_b,n)} + \sum_j \frac{C_{ii}(p_j,r_a,r_b).W(p_j,u_k)}{1+\log C_{ii}(p_j,r_a,n)} + \sum_j \frac{C_{io}(p_j,r_a,r_b).W(p_j,u_k)}{1+\log C_{io}(p_j,r_a,n)}} \tag{5.2}$$

5.5.2 Personalized Similarity of Literals

The second Similarity Method of our approach is a lexical similarity based on literal values that we developed to accomplish the specific goal *SG4: Propose methods to leverage the semantics of literal properties in LOD-based semantic similarity* defined in Chapter 1. It is called Personalized Similarity of Literals because it is applied after the personalization step. In this Subsection, we first describe how the similarity of string literals is calculated and then how the similarity of numerical literals is calculated. For clarity, a running example using the DBpedia resources *dbr:São_Paulo* and *dbr:Rio_de_Janeiro* is used along this Subsection.

From this point forward, please consider *dbr:* as the prefix for <http://dbpedia.org/resource/> and *dbo:* as the prefix for <http://dbpedia.org/ontology/>. For example, *dbr:São_Paulo* is the shortened form for <http://dbpedia.org/resource/Sao_Paulo> and *dbo:populationMetro* is the shortened form for <http://dbpedia.org/ontology/populatio

n_Metro>. Also, another well-known prefix is used in this text to shorten URIs of Linked Data elements: *rdfs:* is the prefix for the RDF schema at <https://www.w3.org/1999/02/22-rdf-syntax-ns>; *dbc:* is the prefix for <http://dbpedia.org/resource/Category>, and *dct:* is the prefix for <http://purl.org/dc/terms/> — from the Dublin Core Metadata Initiative at <http://dublincore.org/>.

Table 5.12: Some properties of the cities São Paulo and Rio de Janeiro.

Property	**Value**	
URI	dbr:São_Paulo	dbr:Rio_de_Janeiro
rdfs:label	São Paulo	Rio de Janeiro
dbo:abstract	São Paulo is a municipality located in the southeast region of Brazil[...]	Rio de Janeiro, or simply Rio, is the second-most populous[...]
dbo:populationMetro	21,090.791	12,280.702
dbo:PopulatedPlace/areaMetro	7,943.752	4,557.343
dbo:PopulatedPlace/areaTotal	1,522.913	1,260.029
dbo:foundingDate	25-01-1554	01-03-1565

Table 5.13: Minimum and Maximum values of some properties.

Property	**Min Value**	**Max Value**
dbo:populationMetro	0	66,400.000
dbo:PopulatedPlace/areaMetro	0	12,012.600
dbo:PopulatedPlace/areaTotal	-1.0	101,475.000
dbo:foundingDate	02-03-0001	01-06-2994

5.5.2.1 Similarity of String Literals

As previously discussed, plain literals are usually represented by Strings. They describe resources to which they belong. In Table 5.12, the property *dbo:abstract* connects the resource *dbr:São_Paulo* to its object (the RDF literal of type String) *"São Paulo is a municipality located..."*.

For the similarity calculation, Strings are treated as bags of words, which means that the order of words is forsaken and only their frequencies, i.e. the number of times they appear in the String, are relevant. These Strings are represented as vectors that store the importance of each word, thus creating the Vector Space Model (MANNING; RAGHAVAN; SCHüTZE, 2008). The similarity between vectors is calculated using the Cosine Similarity measure as shown in Equation 5.3,

$$sim_{string}(A_i, B_i) = \frac{\sum_{j=1}^{n} A_{ij} B_{ij}}{\sqrt{\sum_{j=1}^{n} A_{ij}^2}\sqrt{\sum_{j=1}^{n} B_{ij}^2}} \tag{5.3}$$

Where the numerator corresponds to the dot product of the vectors A_i and B_i, which represent the value of some property i of the resources A and B; and the denominator is the product of their Euclidean Length (MANNING; RAGHAVAN; SCHüTZE, 2008), where A_{ij} and B_{ij} are the terms that compose A_i and B_i.

5.5.2.2 Similarity of Numeric Literals Numeric literals in Linked Data are numbers or dates stored in character Strings. They are used to quantify or date a particular property of one resource. Examples include a movie run time, the population of a city, and the publication date of a book. In Table 5.12, the property *dbo:populationMetro* connects the resource *dbr:São_Paulo* to a numeric literal of value 21,090.791.

Because numeric literal values may vary in scale and magnitude, it is necessary to normalize them among the range values 0 to 1. A movie run time, for instance, can be 10 minutes if the movie is a short take or 3 hours if it is a documentary. For instance, animals' height may vary from a few centimeters to 4 meters. In order to ensure that properties assigned with higher numeric values will not bias the calculation of similarities over smaller ones, we normalize them.

As an example, for the property *dbo:populationMetro* in Table 5.12 to be normalized, we need to retrieve the min and max values of all occurrences of the property. We do this work through a simple SPARQL query. Proceeding with the example, the values retrieved for property *dbo:populationMetro* are 0 and 66,400.000 respectively, as seen in Table 5.13. With these min and max values, a linear normalization process can be applied to each numeric property p through Equation 5.4,

$$normalize(p) = \frac{p - min_p}{max_p - min_p} \tag{5.4}$$

Where min_p and max_p represent the minimum and maximum extreme values of a certain property p. Applying Equation 5.4 to the values of *dbo:populationMetro* showed in Table 5.12 and Table 5.13, the resulting normalized values are 0.3176 and 0.1849 for São Paulo and Rio de Janeiro respectively (as shown in Table 5.14). Similarly, we can apply the same process to other quantifiable values. However, some data types require extra steps to be normalized. For instance, dates must first be converted to type *double* to undergo the same normalization process. After normalization, the similarity of two numeric literals is equal to the absolute value of the difference between them, subtracted from 1, as shown in Equation 5.5,

$$sim_{numeric}(A_i, B_i) = 1 - |A_i - B_i| \tag{5.5}$$

where A and B are resources and A_i and B_i are the normalized values of property i for each of them, respectively.

5.5.2.3 The Combined Similarity Function for Literals Having shown how to calculate the similarity of String and numeric literals, we combine them in a single similarity function. Before, we must remember that the similarity only applies to properties that are shared by two resources. It makes no sense, for instance, to compare the *age of a person* to a *country area*. Formally, and for the sake of simplicity, we summarise Equations 5.5 and 5.3, i.e. the similarity of one particular property i shared by resources

Table 5.14: Normalized values of some properties presented in Table 5.12.

Property	**Normalized Value**	
URI	dbr:São_Paulo	dbr:Rio_de_Janeiro
dbo:populationMetro	0.3176	0.1849
dbo:PopulatedPlace/areaMetro	0.0006	0.0003
dbo:PopulatedPlace/areaTotal	0.0001	0.0001
dbo:foundingDate	0.5188	0.5225

A and B in Equation 5.6,

$$sim_{property}(A_i, B_i) = \begin{cases} sim_{numeric}(A_i, B_i) \\ sim_{string}(A_i, B_i) \end{cases} \tag{5.6}$$

where $sim_{numeric}(A_i, B_i)$ is calculated if Numeric, and $sim_{string}(A_i, B_i)$ if String.

The combined similarity for *all* shared properties of two resources A and B is then given by Equation 5.7,

$$sim_{combined}(A, B) = \frac{\sum_{i=1}^{N} sim_{property}(A_i, B_i)}{N} \tag{5.7}$$

where N is the number of shared properties between the resources A and B. Given the similarity scores obtained for each property, shown in Table 5.15, the combined similarity score obtained in Equation 5.7 is 0.926.

Table 5.15: Similarity scores of each property from Table 5.12.

Property	**Similarity Score**
rdfs:label	0
dbo:abstract	0.767
dbo:populationMetro	0.867
dbo:PopulatedPlace/areaMetro	0.999
dbo:PopulatedPlace/areaTotal	0.999
dbo:foundingDate	0.996

5.5.2.4 Penalizing Sparsity The information available in the LOD cloud is not always complete. The same triples may not describe resources of the same type. For example, the property *dbo:ethnicGroupsInYear* in the triple <dbr:Brazil,dbo:ethnicGroupsInYear,2010-01-01^^(xsd:date)> does not apply for *dbr:Chile* but *dbr:Peru*, being all three countries of the same type *dbo:PopulatedPlace*. As a consequence, resources may be compared unfairly. It turns out that the similarity score of two resources that compare several properties is likely to be lower than two other resources sharing one single property, considering here the worst-case scenario. It seems unfair therefore to penalize a resource that is highly described over those which share only a few properties.

For instance, in a scenario where the similarities of items A, B, and C are being estimated, and A and B share three properties, while B and C share five. If those three properties shared by A and B are closely related, while B and C also have those same properties having the same similarity score, but another two with very different values, then A and B would be deemed more similar then B and C, even though their similarities regarding three of their attributes are identical.

In order to avoid such an inconvenience, the combined similarity function is penalized according to Equation 5.8,

$$penalty(A, B) = \frac{|A \cap B|}{|A| + |B|} \tag{5.8}$$

where $|A \cap B|$ is the number of shared properties of A and B, and $|A|$ and $|B|$ are the number of properties of the most specific class of A and B, respectively. The penalization is weighed by the similarity score obtained in the Equation 5.7,

$$sim_{literals}(A, B) = sim_{combined}(A, B) \cdot penalty(A, B) \tag{5.9}$$

The most specific class of the resources in the running example is <http://dbpedia.or g/ontology/PopulatedPlace>, which has 112 associated datatype properties. Given that only five were used in the example, the final similarity score based on literals is 0.041, a big difference from the unpenalized score of 0.926.

5.5.2.5 The Literals Similarity (LiSim) Literals Similarity (LiSim) combines the similarity score obtained from comparing literals with a pure link-based method LB. This step of the metric can be done by using any LOD-based similarity measure. In Chapter 6, after calculating all the literal similarities, we combine the outcomes with two link-based baseline methods: LDSD and ReSim, which means that LB is substituted by LDSD or ReSim equations. The final LiSim structure is shown in Equation 5.10,

$$\begin{aligned} HSLD(A, B) = {} & \alpha \cdot sim_{literals}(A, B) + \\ & (1 - \alpha) \cdot (1 - LB(A, B)) \end{aligned} \tag{5.10}$$

Where α is a coefficient between 0 and 1, which determines the percentage of influence of each similarity method on the final similarity score. This means that an $\alpha = 0$ is equal to use only LB, and an $\alpha = 1$ is equal to use only the similarity of literals. When LB is a distance function, it is subtracted from 1. Otherwise, no subtraction is needed.

5.6 SUMMARY

This chapter explains the proposed solution, starting with the background information supporting the proposed branches and expounding each branch individually. Figure 5.1 showed each branch as a step in the whole process, and the following sections detailed each step. Step 1 refers to a feature selection strategy that filters a subset of the most predictive properties to a particular knowledge domain. In Step 2, we enrich the user model with DBpedia resources and automatically weigh the features by exploring the user's previous preferences. Finally, Step 3 runs a recommendation model using a link or literal-based similarity measure over the input personalized dataset, resulting in recommendations that fit user preferences.

The next chapter describes the implementations and execution of the experiments which objective is to evaluate the proposed methods, by comparing results using the personalized user model versus results using only the baseline methods.

Chapter

6

EXPERIMENTAL EVALUATION

This chapter presents the experiments that evaluate our proposal. We highlight the characteristics of the datasets, methodologies employed, and results obtained. In addition, we discuss the results considering the Research Questions proposed in Chapter 1.

6.1 INTRODUCTION

The evaluation phase of this research is composed of three experiments as follows:

- E1: The first experiment is called the Literals Similarity (LiSim) experiment and consists of analyzing the similarity between two resources in a LOD graph, considering the literal properties instead of only the link properties. We compare the LiSim measure to pure-link approaches using two baseline methods, Linked Data Semantic Distance (LDSD) and Resource Similarity (ReSim). Also, we test our methodology against two diverse domains: movies and music. Considering the information stored in RDF literals, we expect comparisons between Linked Data resources to be estimated more precisely.

- E2: The second experiment is called Personalized Linked Data Semantic Distance (PLDSD), in which we personalize the user model by weighting the properties according to the user's past choices in the system. We describe a generic model and use LDSD as the baseline method to compare it with our personalized approach. Moreover, any semantic measure based on links can be used with this personalized method by only modifying the last step of the architecture (see Section 5.1), where the baseline semantic similarity metric is added to the recommendation model.

- E3: The third experiment is called Summarization, and its goal is to validate the hypothesis that filtering the most relevant links before the step of user model personalization can lead to more effective recommendations in Linked Data-based systems. To prove this, we implement a Feature Selection (Step 1 in Figure 5.1) strategy that filters the K most relevant properties and subsequently weighs and ranks each property in the personalization step with the PLDSD method.

The complete project has been available on an Anonymous Github under the URL <https://anonymous.4open.science/r/lodweb-pldsd-40BD/>. Researchers interested in replicating and conducting further comparative studies can find the necessary documentation at this location. In the next sections, we describe the setups of datasets, methodology, and metrics used, and then we discuss each experiment in detail.

6.2 DATASETS SETUP

We use two databases in the experiments: the Movielens 1M dataset[1] and the Last.fm dataset[2]. Movielens 1M has 1 million ratings from 6000 users on 4000 movies (1-5 stars rating). The density of ratings in the dataset is 4.26%. Based on it, we model one data model capable of storing the user model, i.e., users, movies, and user ratings over movies, and also can store the content of movies. Then, we import MovieLens data mapped to DBpedia into the database using MappingMovielens2DBpedia[3], which provides RDF identifiers — URIs — for each movie on MovieLens 1M. Therefore, this initial setup allows us to access the resources and all their connections from DBpedia through SPARQL queries online. While the algorithm is running, property weights per user and results of similarity measures are also stored. This allows us to set up parameters and to compare results.

The second dataset is the Last.FM Million Song[4]. It contains listening information from almost 2 thousand users, about approximately 1 million songs. Nonetheless, we take the listening data summarized by the user-artist for this work rather than considering the songs played. For this reason, during the dataset setup step, we used another set of data released during the HETRec 2011 Workshop and published as another project of raw data mapping to Linked Data[5]. As is in the movie mapping project, this one provides DBpedia URIs for each musical artist or band in the Last.FM dataset.

6.3 HARDWARE SETUP

The experiments were carried out on a machine with the following configuration.

- CPU: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz 1.99 GHz
- Memory: 16.0 GB
- Storage: Kingston 512 GB SSD
- Operating System: Windows 10 Home 64 bits

[1]https://grouplens.org/datasets/movielens/1m/

[2]https://www.last.fm/

[3]https://github.com/sisinflab/LODrecsys-datasets/blob/master/Movielens1M/MappingMovielens2DBpedia-1.2.tsv

[4]<http://millionsongdataset.com/lastfm/>

[5]<https://github.com/sisinflab/LinkedDatasets/blob/master/last_fm/mappingLinkedData.tsv>

6.3.1 Software tools

On top of the aforementioned hardware configuration we use the following software tools.

- JAVA and JENA: According to W3C [6] "Apache Jena is a Java framework to construct Semantic Web Applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL, and includes a rule-based inference engine."
- MySQL: Relational database management system
- Apache Jena Fuseki: open-source RDF graph database (triplestore) that complies with the SPARQL 1.1 specification and supports SPARQL query
- Apache Tomcat: Free and open-source HTTP web server environment

6.3.2 Limitations

The 8th Generation Intel® Core™ i7 Processor used in the experiments has 4 cores with an 8 simultaneous threads capability. The Processor base frequency is 1.80 GHz with 4.00 GHz of Turbo Boost technology frequency. Although this seems to be an acceptable hardware configuration, the online SPARQL queries executed by JENA against the DBpedia database are heavy and require more power from the processor. Having said that, I would point out that, for example, each round of experiments in experiment 2 took three days to be completed.

6.4 METHODOLOGY

For each experiment, we simulate Movie and Music Recommender Systems that retrieve Top-N recommendations to the user on the respective domains.

Given that the user preferences on the MovieLens dataset are expressed in a 5-star Likert scale, some research that uses these datasets put effort into making the user ratings binary (NOIA et al., 2012; MUSTO et al., 2016). In the first experiment (E1: LiSim), for simplification, we also adopted the binary representation of the user profile. That means we build the training set based on all the items the user liked (has given positive feedback).

Although, in the second experiment (E2: PLDSD), we decided to maintain the original 5-star scale and consider 3 different scenarios to represent the users' positive feedback: $PLDSD_{rat=5}$ means that only ratings equal to 5 are considered; $PLDSD_{rat>=4}$ considers besides movies rated 5, those which the user has rated 4; and $PLDSD_{rat>=3}$ takes ratings equal to 5, 4 and 3 as positive feedback. This strategy allows us to compare results and adjust our methodology better.

Due to this configuration, we perform the PLDSD experiment 3 rounds for each user in the dataset. First, we isolate the user model according to the current strategy — $PLDSD_{rat=5}$, $PLDSD_{rat>=4}$, $PLDSD_{rat>=3}$. In other words, we take the positively

[6] <https://www.w3.org/2001/sw/wiki/Apache_Jena>

rated movies, according to what positive feedback means, and build the training or known data dataset. Then, we randomly take a subset of 200 movies (discarding those already in the user model subset) and build the test dataset.

Similarly, in the Last.FM dataset, user preferences about artists are expressed through the number of times each user has listened to their songs. Consequently, the concept of positive feedback for this work had to be constructed by isolating an expressive number representing a user listening to a well-liked artist throughout her daily routine. After statistical analysis, we chose the number 500 to represent positive feedback; therefore, artists' tracks heard less than 500 times are considered negative feedback was given by a user. Based on this premise, we built user models from the set of items positively evaluated by each user.

For the third experiment (E3: Summarization), we considered only 5-star movies to calibrate the MovieLens training dataset. The previous experiment demonstrated that the $PLDSD_{rat=5}$ strategy performed the most accurate results for all scenarios. For this reason, we adapted our methodology to consider rating 5 as positive feedback and all the other ratings — 1, 2, 3, and 4 — as negative feedback. However, we perform tests several times where we vary the k parameter to verify the extent to which the quantity of features filtered influences the personalization results.

After running the training steps, we perform the testing step by running the algorithms according to which experiment is being tested (E1, E2, or E3). We also run the same dataset configuration using two baseline methods — LDSD and ReSim — to compare the improvements achieved. The user model also plays the role of validation dataset to which we compare the ranked result list coming from the recommendation method.

Moreover, we performed some statistical tests to assess the significance of the results. First, we run the chi-squared goodness-of-fit test over the ranked list of movies. As this test revealed the normal distribution of the data, we opted to apply a paired T-test using a p-value $<< 0.0001$ to which each method is tested against its correspondent baseline method (LDSD or ReSim) results from the sample. As the sample length varies according to how many movies or artists the user has rated, we ran the statistical tests for each user tested.

6.5 METRICS

Having the ranked list of recommended items (movies / musical artists and bands) and the predefined training dataset of liked items, three evaluation metrics are applied to assess the quality of the generated ranks. These metrics consider how well the similarity algorithm can predict the desired set of similar items. Precision@K, Mean Average Precision (MAP), and Normalized Discounted Cumulative Gain (NDCG) are chosen metrics. They were explained in Chapter 3.

6.6 E1: LITERALS SIMILARITY (LISIM)

We first present the results in the movie domain and then in the music domain. Figures 6.1 and 6.3 present the evaluation results of LiSim in each domain using LDSD

(LISIM+LDSD) with variations of α. Results for $\alpha = 0$ mean that LDSD is the only similarity measure applied. Alternatively, $\alpha = 1$ indicates the plain analysis of literals. Figures 6.2 and 6.4, present the evaluation results of LiSim using ReSim (LISIM+RESIM) with variations of α. Results for $\alpha = 0$ means that RESIM is the only similarity measure applied, while $\alpha = 1$ means that the literal similarity is considered. The values highlighted in all the graphs are the highest statistically significant ones (see Section 6.4 for more details on the statistical tests).

6.6.1 Results from the Movie domain

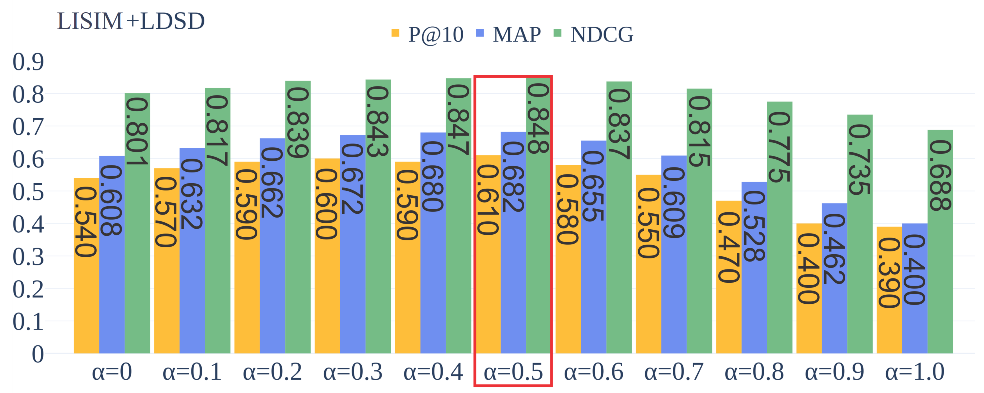


Figure 6.1: LISIM+LDSD evaluation results grouped by metrics from the movie domain.

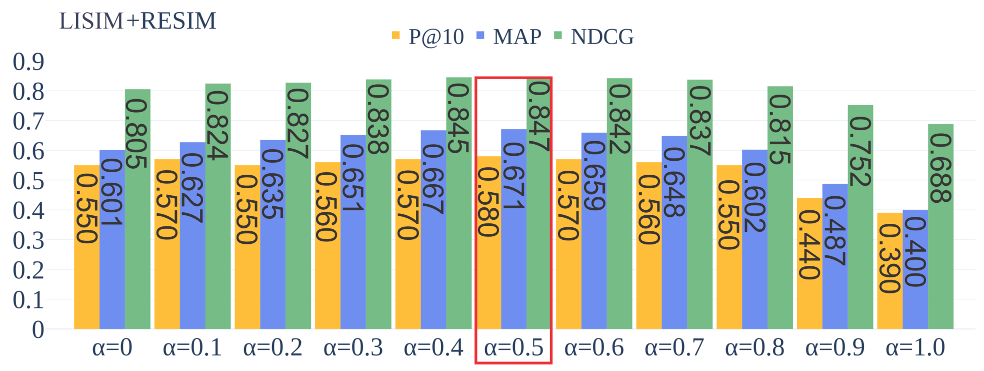


Figure 6.2: LISIM+RESIM evaluation results grouped by metrics from the movie domain.

Figure 6.1 presents the LISIM+LDSD results while Figure 6.2 presents LISIM+RESIM results. Considering the P@10 results, which are represented by the yellow bars, we can observe that LISIM+LDSD outperforms LISIM+RESIM where $0.2 \leq \alpha \leq 0.6$. On the other hand, LISIM+RESIM outperforms LISIM+LDSD for $0.7 \leq \alpha \leq 0.9$. With $\alpha = 0$, i.e. when the literal analysis is not considered, the ReSim method slightly outperforms LDSD.

The highest precision values for both approaches occur when $\alpha = 0.5$. This means precision at the top 10 results is higher when both literals and links are considered equally. Moreover, when $\alpha = 0.5$, the LiSim measure outperforms LDSD in 13% and also outperforms ReSim in 5.5%. It is important to point out that even for lower values of α, the P@10 rates outperform the ones achieved by each pure link-based measure ($\alpha = 0$) in both combinations LISIM+LDSD and LISIM+RESIM.

In Figure 6.1, all results in the interval $0.1 \leq \alpha \leq 0.7$ are higher than $\alpha = 0$ value, i.e., LDSD $= 0.54$; and in Figure 6.2, this interval is even greater, $0.1 \leq \alpha \leq 0.8$, if we consider the values greater than or equal to $\alpha = 0$ value, i.e., ReSim $= 0.55$.

As to the MAP results, as shown in Figures 6.1 and 6.2, we observe similar findings as P@10. In particular, the highest MAP results occur when $\alpha = 0.5$, where LiSim outperforms LDSD in 12.2% and outperforms ReSim in 11.6%. The NDCG results shown in Figures 6.1 and 6.2 also report similar remarks to those observed in MAP and P@10 results. The best NDCG results are also observed when $\alpha = 0.5$, where they outperform the link-based methods. In particular, LiSim outperforms LDSD in 5.9%, and LiSim outperforms ReSim in 5.2%.

6.6.2 Results from the Music domain

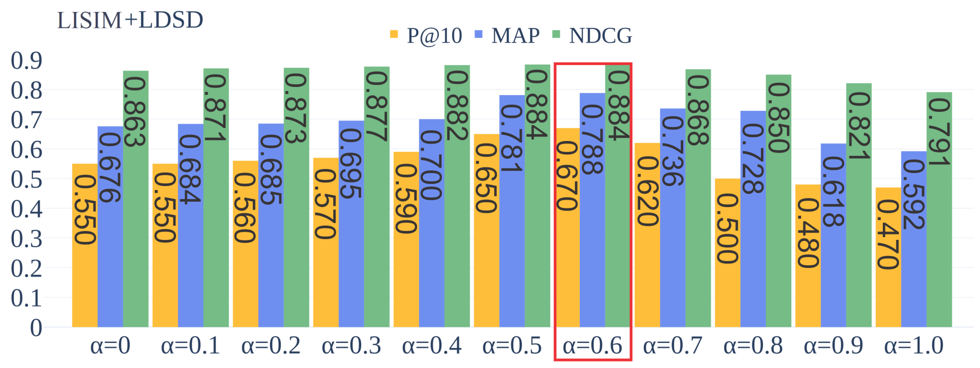


Figure 6.3: LISIM+LDSD evaluation results grouped by metrics from the music domain.

The results shown in Figures 6.3 and 6.4 point out the highest P@10 values for both LISIM+LDSD and LISIM+RESIM occur when $\alpha = 0.6$. This indicates that literals can have greater influence than the movie domain. There, the best results are evidenced when both literals and links are considered equally, i.e. when $\alpha = 0.5$. In particular, for $\alpha = 0.6$, LiSim outperforms LDSD in 21.8% and outperforms ReSim in 16.1%, both greater than the in the movie domain.

As to the MAP and NDCG results, shown in Figures 6.3 and 6.4 , we also observe that the results are similar to the results achieved in the movie domain. The highest MAP results occur when $\alpha = 0.6$, where LiSim outperforms LDSD in 16.7% and outperforms

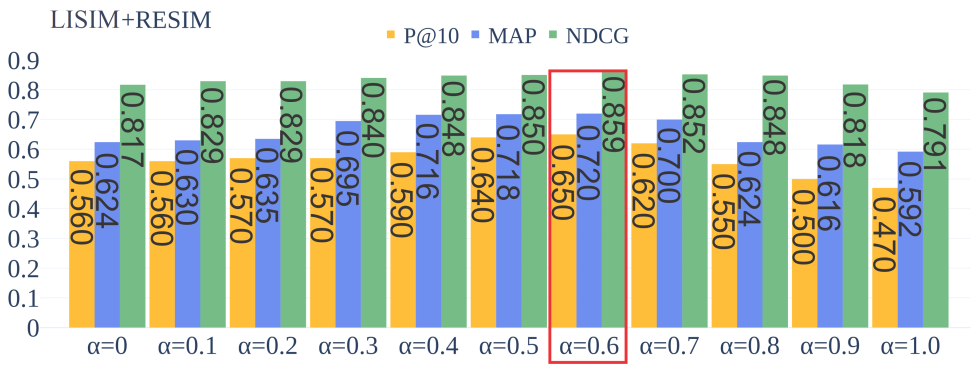


Figure 6.4: LISIM+RESIM evaluation results grouped by metrics from the music domain.

ReSim in 15.5%. Considering the NDCG results, for $\alpha = 0.6$, LiSim outperforms LDSD in 2.4% and outperforms ReSim in 5.2%. These increases are lower than the ones evidenced in the movie domain. This can be explained because when NDCG reaches 90%, it becomes more difficult to augment the already high results. It was an expected behavior of the NDCG metric observed empirically while running our experiments with different setups.

6.6.3 Discussion and Contributions

This experiment intended to respond to *Q1: Can the system be more precise in recommending items to the user if it calculates the similarity using both literal properties and link properties*?

The evaluation results show that the precision of the movie and music recommendations improves when the similarity of literals is considered. The Literals Similarity (LiSim) performs better when both link and literal-based similarities are considered. This highlights the potential of other LOD-based approaches using the information in RDF literals.

As to the similarity measures, we have evidence that both the similarity measures, LDSD and ReSim, benefit from using literals. In the movie domain, the ReSim measure achieves slightly better results than the LDSD when $\alpha = 0$, i.e., when literals are not considered. On the other hand, in the music domain, we observe that LDSD achieves the best results for MAP and NDCG metrics. This analysis, however, is based solely on the object properties, i.e., the links between resources.

We also note that the results behave differently when the link-based similarity measures are combined with LiSim. The results show that LISIM+LDSD slightly outperforms LISIM+RESIM in both domains. In the scenario when $\alpha = 0.5$, LISIM+LDSD outperforms LISIM+RESIM in 5.2% for P@10 and in 1.6% for MAP in the movie domain. Likewise, in the music domain, when $\alpha = 0.6$, LISIM+LDSD outperforms LISIM+RESIM in 3.1% for P@10; in 9.4% for MAP, and in 3% for NDCG.

This is more evident when we compare the results in both domains regarding α variations. For instance, the distribution of the curves' growth rate for the music graphics is smoother than those in the movie domain. We ran exploratory SPARQL queries better to know the graph characteristics of the RDF datasets. Thus, the difference between the α optimum values might be explained because resources from the music domain have several links and literals more balanced than those from the movie domain. For instance, the difference between the number of links and literals in the music domain is 121, and in the movie domain is 152.

The overall outcome shows that the highest ranking results occur when $\alpha = 0.6$ in the music domain (Figures 6.3 and 6.4), and when $\alpha = 0.5$, for the movie domain. This suggests that the ranking quality of the music recommender system suffered more influence of literals than the links. This can be explained by the average number of object properties (links) from the resources comprised in the music candidate dataset (226) against the average number of datatype properties (105). For the resources of the movie dataset, these numbers are 184 for links and 32 for literals, a more widely distance, denoting a less balanced dataset. Hence, the distribution of the results in the music domain is expected to differ from the movie domain, as the ignored properties (the literals) amount varies from one domain to another.

Considering the aforementioned considerations, we conclude that *Q1: Can the system be more precise in recommending items to the user if it calculates the similarity using both literal and link properties?*, was positively answered, and the Specific Goal *SG1: Propose a feature selection approach to filter relevant properties according to the domain*, was achieved.

6.7 E2: PERSONALIZED LINKED DATA SEMANTIC DISTANCE (PLDSD)

This section presents the results of each previously chosen precision metric on the graph personalization methodology discussed in Chapter 5, Section 5.4. The values highlighted in all the results tables are the highest statistically significant ones (see Section 6.4 for more details on the statistical tests).

6.7.1 Results from the Movie domain

In this experiment, we use only the movie dataset and test the three taken user model strategies: $PLDSD_{rat=5}$, $PLDSD_{rat>=4}$ and $PLDSD_{rat>=3}$. We also show results on LDSD as a baseline method of comparison. As results are different for each user, we take 3 of them who have given different quantities of ratings to movies and summarize their results in Tables 6.1, 6.2 and 6.3. The goal is to show examples of outcomes and to provide a source of comparison.

Table 6.1 shows results from User #1, who has previously given 40 ratings on movies. Results demonstrate that PLDSD using only movies rated as 5 stars performs better in modeling the user's preferences than PLDSD using ratings greater than or equal to 4 and 3, for every metric analyzed. Also, both $PLDSD_{rat=5}$ and $PLDSD_{rat>=4}$ values overcome the baseline method LDSD. We ran a paired T-test using $p << 0.0001$ and found a significant improvement considering ratings equal to 5, and ratings greater than or equal to 4. We observed that even ratings greater than or equal to 3 have statistical significance compared to the baseline method LDSD.

Table 6.1: Results from each metric of PLDSD for user #1 who has rated 40 movies. LDSD is used as the baseline method.

	Precision@5	Precision@10	MAP	NDCG
$LDSD$	0.520	0.504	0.608	0.801
$PLDSD_{rat=5}$	**0.776**	**0.700**	**0.729**	**0.859**
$PLDSD_{rat>=4}$	0.698	0.666	0.608	0.853
$PLDSD_{rat>=3}$	0.281	0.266	0.320	0.556

Table 6.2: Results from each metric of PLDSD for user #2 who has rated 130 movies. LDSD is used as the baseline method.

	Precision@5	Precision@10	MAP	NDCG
$LDSD$	0.514	0.516	0.700	0.824
$PLDSD_{rat=5}$	**0.710**	**0.788**	**0.731**	**0.873**
$PLDSD_{rat>=4}$	0.654	0.699	0.600	0.802
$PLDSD_{rat>=3}$	0.274	0.257	0.318	0.551

Table 6.3: Results from each metric of PLDSD for user #3 who has rated 246 movies. LDSD is used as the baseline method.

	Precision@5	Precision@10	MAP	NDCG
$LDSD$	0.444	0.432	0.667	0.831
$PLDSD_{rat=5}$	**0.813**	**0.813**	**0.764**	**0.889**
$PLDSD_{rat>=4}$	0.722	0.753	0.623	0.867
$PLDSD_{rat>=3}$	0.300	0.298	0.320	0.588

In a similar analysis, Table 6.2 corroborates the previous results by investigating User #2, who has given 130 ratings on movies. Likely, the best picture is when only 5-star rated movies are considered by the user model. Here we can notice some variation in which the k value of Precision performs better. Despite $Precision@5$ performs better overall, at $PLDSD_{rat=5}$ and $PLDSD_{rat>=4}$, $Precision@10$ overcomes it. Comparing results' values in Table 6.2 to Table 6.1, we conclude that augmenting movies in the user model can be related to better results. Results from paired T-tests using $p << 0.0001$ also showed statistical significance.

Finally, Table 6.3 shows the results from User #3, who has given more movie ratings: 246. Similar analysis as the previous tables about User #3, including the statistical significance, can be done. As expected, $PLDSD_{rat=5}$ results overcome other strategies.

6.7.2 Discussion and Contributions

In this experiment, we validate the hypothesis that LOD-based Recommender Systems can benefit from a personalized feature ranking method. This was stated in Chapter 1 by *Q2: Can the system recommend items better suited to the user's taste by exploiting her preferred properties from past choices?*

A statistically significant improvement ($p << 0.0001$, assessed through paired T-tests) was obtained for every user model strategy. In other words, it shows that leveraging the users' past ratings as criteria to rank the more significant features to recommend items to the user is statistically significant (it did not occur randomly) compared to the baseline method LDSD.

The overall picture leads us to conclude that the proposed method overcomes the baseline method on $PLDSD_{rat=5}$ and $PLDSD_{rat>=4}$ to every metric. The same conclusion cannot be made when we take ratings equal to 3 to compose the training dataset: all results are lower than the baseline method in this picture. Although, the advantage of taking ratings equal to 3 into account is that it expands the user model with more movies. For example, in Table 6.3, from a set of 246 rated movies, we go to a set of 206 when we take ratings 5, 4, and 3, to a set of 136 movies when we take ratings 5 and 4 and to only 52 movies when we take just ratings equal to 5. Nonetheless, as a consequence, the training set becomes too diverse that it is hard to find a model that fits a particular user. Thus, this explains the weak results found for $PLDSD_{rat>=3}$ in every table.

Another conclusion is that the more the user has rated movies, the better the results will be on each metric (comparing highlighted values among tables). It occurs because the number of properties analyzed increases according to the number of liked movies. Taking the $PLDSD_{rat=5}$ model as example, User #1, has given positive feedback to 7 movies; User #2, to 42 movies; and User #3 to 52 movies. Hereupon, User #1 had 33 properties analyzed, against 56 from User #2 and 87 from User #3. Due to this 164% increase in the number of properties between User #1 and User #3, metrics also had significant increases: $Precision@10$ increased by 16%, MAP increased by 5%, and NDCG increased by 3%.

Finally, we find three research questions stated in Chapter 1 were positively responded to by this experiment, which are *Q2: Can the system recommend items better suited to* *the user's taste by exploiting her preferred properties from past choices?*; *Q3: Can we automatically pick the properties that most influence the user's choices regardless of the domain?*; and *Q4: Is the proposed link ranking method feasible to personalize the system?*

In addition, we demonstrated that two Specific Goals were achieved in this experiment: *SG2: Propose a personalization methodology that weighs links in a LOD graph based on the user's past ratings on the items in a recommender system*; and *SG3: Propose a user profile modeling from the personalized graph obtained in the previous step.*

6.8 E3: SUMMARIZATION

This section presents and discusses the results from Experiment 3, considering the methodology and objectives proposed in this work. Before the tests, we built the user models from the set of items positively evaluated by each user. The concept of positive rating was constructed differently for each domain dataset, movies, and music, according to the methodology explained in Section 6.4.

The personalization step results in lists of ranked properties used to feed the recommender system, which runs two similarity algorithms: LDSD and Personalized LDSD (PLDSD). A k value may limit the ranked lists due to the feature selection preprocessing step. As we consider 3 k values for each strategy in addition to the turn without limiting the value of k, and 2 similarity strategies, the experimental evaluation consists of 8 rounds of testing for each user on the movie dataset and another 8 rounds for each user on the music dataset.

In order to facilitate the understanding of the results tables, we use the acronym *LDSD* to represent the recommender model built under plain LDSD, and the acronym *PLDSD* to represent the recommender model that considers our personalized approach, as presented in Section 5.4. Tests that consider the feature selection step are identified by the value assigned to k in the result tables: Table 6.4 and Table 6.5. For example, PLDSD k=10 means that results encompass the personalization method and the preprocessing step that selects the 10 most relevant properties. On the other hand, LDSD k=0 demonstrates the results from when neither customization nor feature summary is applied.

6.8.1 Results from the Movie domain

Table 6.4 summarizes the average (Avg) results of a hundred users from the movie dataset, considering the LDSD scenarios with and without feature selection and PLDSD with and without feature selection. In the movie scenario, each of the 100 user models comprises at least 7 and, at most 52 positively rated movies. The test set is composed of 200 not evaluated movies. The value of k varies from 10 to 535, which is the maximum number of properties for the movie domain in DBpedia. LDSD and PLDSD without feature selection (k=0) are used for comparison purposes.

Table 6.4 shows a statistically significant increase (see Section 6.4 for more details on the statistical tests) on all metrics when applying the PLDSD with k=10. The LDSD approach presents worse results than personalized and filtered rounds.

Table 6.4: Results from the movie dataset considering diverse scenarios.

Strategy	**Precision@5**	**Precision@10**	**MAP**	**NDCG**
LDSD k=0	0.450	0.450	0.501	0.743
LDSD k=10	0.500	0.450	0.565	0.798
LDSD k=100	0.450	0.425	0.550	0.751
LDSD k=535	0.450	0.425	0.499	0.744
PLDSD k=0	0.450	0.450	0.511	0.757
PLDSD k=10	**0.550**	**0.450**	**0.582**	**0.809**
PLDSD k=100	0.550	0.450	0.581	0.794
PLDSD k=535	0.550	0.425	0.504	0.783

6.8.2 Results from the Music domain

Table 6.5 summarizes the average (Avg) results of a hundred users from the music dataset, considering LDSD scenarios with and without feature selection and PLDSD with and without feature selection. Each of the 100 user models in the music scenario consists of at least 14 and at most 50 positively rated artists. The test set is composed of 200 other not evaluated artists. The value of k varies from 10 to 512, the maximum number of properties for the music domain in DBpedia. LDSD and PLDSD without feature selection (k=0) are used for comparison purposes.

As with the movie domain, the results from the music context shown in Table 6.5 also achieved higher significant values (see Section 6.4 for more details on the statistical tests) when applying PLDSD with k=10. And again, the results from LDSD without personalization and filters show worse values among the rounds of experiments.

Table 6.5: Results from the music dataset considering diverse scenarios.

Strategy	**Precision@5**	**Precision@10**	**MAP**	**NDCG**
LDSD k=0	0.433	0.450	0.491	0.622
LDSD k=10	0.462	0.450	0.520	0.671
LDSD k=100	0.425	0.425	0.503	0.621
LDSD k=512	0.420	0.425	0.488	0.603
PLDSD k=0	0.436	0.450	0.509	0.635
PLDSD k=10	**0.470**	**0.556**	**0.535**	**0.731**
PLDSD k=100	0.451	0.445	0.508	0.647
PLDSD k=512	0.450	0.440	0.500	0.620

6.8.3 Discussion and Contributions

This experiment was intended to respond to *Q5: Can the system benefit from a preprocessing step that filters the domain-relevant features before entering the user personalization method*? These experiments have tested the baseline method LDSD and the personalized approach PLDSD varying the number of resultant features k. Results show significant improvements, as shown below.

Results from the movie dataset experiments reveal that MAP and NDCG metrics obtained the best values. Additionally, those values are higher when the similarities are calculated using the feature selection step. Figure 6.5 presents the MAP and NDCG values for both LDSD and PLDSD approaches with different values of k. This graphical representation emphasizes the significant growth of approximately 7% in NDCG for LDSD values and of 8% in the NDCG for PLDSD values. Concerning the MAP metric, experiments achieved 13% growth in the LDSD rounds and 14% in the NDCG for PLDSD values when applying the feature selection step.

The results of Precision@5 and Precision@10 from the movie experiments show positive and negative oscillations between values, highlighting a positive gain of Precision@5 when calculating PLDSD with the feature selection. Furthermore, all metrics values are higher when the number of properties is reduced in the pre-selection step, both for LDSD and PLDSD, especially when $k = 10$.

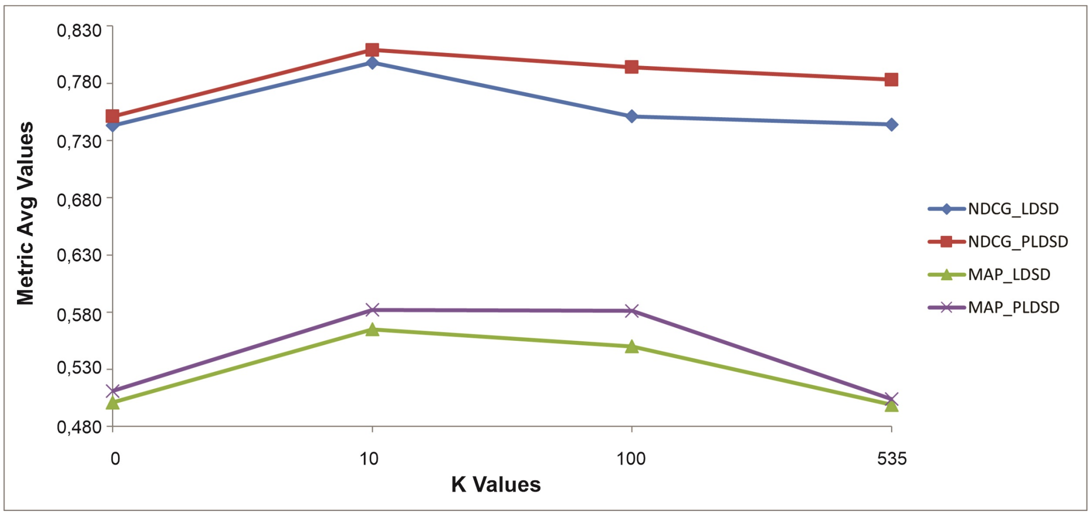


Figure 6.5: NDCG and MAP results *versus* the k value of pre-selected features from the movie domain.

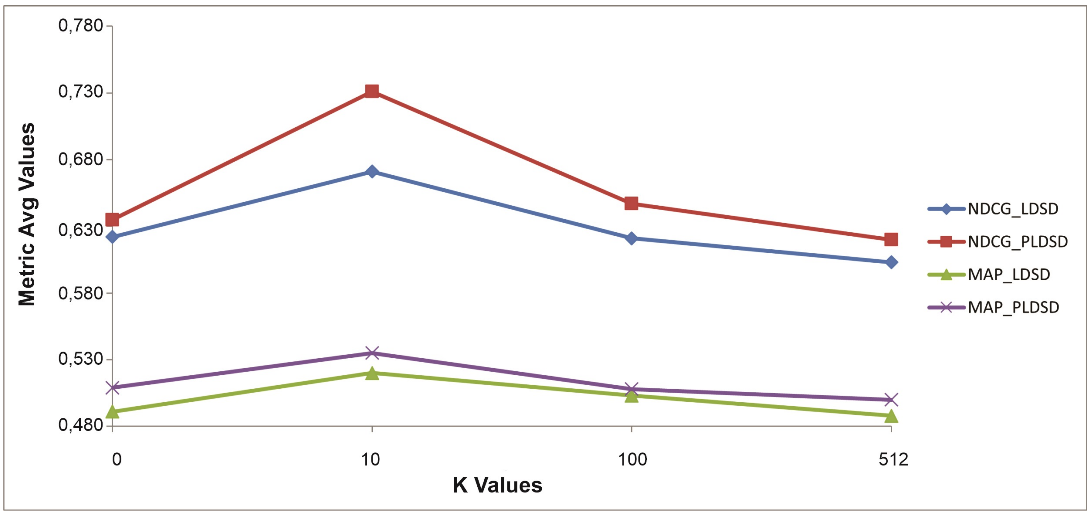


Figure 6.6: NDCG and MAP results *versus* the k value of pre-selected features from the music domain.

The results from the music dataset also showed improvements, although not as expressive as the movie domain, especially for the MAP metric. Figure 6.6 shows a growth of 8% in the NDCG for LDSD values and 15% in the NDCG for PLDSD values. Regarding the MAP metric, experiments achieved 6% growth in the LDSD rounds and 5% in the NDCG for PLDSD values when applying the feature selection step.

Originally, the ABSTAT method behaved differently depending on the selected knowledge domain (NOIA et al., 2018). A comparative analysis leads us to conclude that ABSTAT summaries are strongly grounded in the ontological nature of the knowledge graph. At the same time, our approach emphasizes user preferences drawn from their previous interactions with the system. This means that PLDSD applied to different domains will perform similarly, although it may vary slightly. Nevertheless, the combined use of the two techniques is more efficient than either one separately.

However, further analysis is required to investigate if the nature of the music subject does not allow an accurate prediction of its properties and weights. The maximum number of properties retrieved from DBpedia is similar to domains — 535 and 512 — although the modeling is very different. A musical artist can be linked to many different songs by the property dbo:artist of, while an actor is apt to be linked to fewer movies by the property is dbo:star of. For example, the band The Rolling Stones is linked to over 300 songs, while Anthony Hopkins, the actor with one of the most solid careers, could act in only 137 movies so far.

Conversely, the system's overall performance significantly reduces the processing time, both for the movie and song datasets, since the number of properties being computed is reduced by approximately 80% when considering $K = 10$. Taking the example of the user #1 — who positively rated 246 artists — the computation time was reduced from 240 to 67 minutes by adding the selection step, a reduction of 72%.

This section concludes that *Q5: Can the system benefit from a preprocessing step that filters the domain-relevant features before entering the user personalization method?*, was positively responded to and that *SG4: Propose methods to leverage the semantics of literals properties in LOD-based semantic similarity* was successfully achieved.

6.9 SUMMARY

This chapter elucidated how the research accomplished the Specific Goals defined in Chapter 1. Experimental evaluation is the last step of the methodology and it encompass the implementation and evaluation of the proposed methods by comparing results using the personalized user model versus results using only the baseline methods. We described the datasets, methodology, and metrics employed in three experiments. It also discusses the results obtained, addressing the challenges and the main contributions of each experiment. Next chapter concludes the thesis by describing the improvement points and future work suggestions.

Chapter 7

FINAL REMARKS

This chapter summarizes the main arguments and findings of the research. We also critically reflect on the work's outcomes, highlight the limitations, and suggest areas for future research.

7.1 CONCLUSION

In this work, we proposed personalization strategies for LOD-based recommender systems. The proposal's main goal was to develop methods for calculating Linked Data Semantic Similarity, leading to more accurate recommendations. Throughout this thesis, we show how the specific goals defined in Chapter 1 was implemented and achieved through solution models that led to more accurate recommendation systems.

A solution architecture (see Chapter 5) was developed as a product of the research. It comprises all the thesis proposal steps, starting with the feature selection method. It reduces the database to the properties that most imprint semantics to the knowledge domain in concern (Step 1). Then it is followed by the user model personalization method, which analyses the user's past interactions with the system to rank the properties used in the recommender model (Step 2). Finally, a recommendation model unites the previous steps with one or more similarity measures, either link- or literal-based, to calculate the rank of items for future user recommendations (Step 3).

Our generic architecture allows different semantic approaches to be applied to the recommender system. For example, in the first step, it is possible to use different feature selection techniques, whether based on statistics or expert supervision. In the experiments we built to validate the results of the research, we used the following methods: the Summarization automatized FS method (NOIA et al., 2018) in step 1; the authorial weighting personalization method (SILVA; DURãO; CAPRETZ, 2019) in step 2; and the baseline methods Linked Data Semantic Distance (LDSD) (PASSANT, 2010) and Resource Similarity (ReSim) (PIAO; ARA; BRESLIN, 2016) in step 3.

Before testing, we enriched two well-known recommendation datasets, MovieLens and Last.FM, with linked data from DBpedia. After that, we implemented and ran three experiments intending to respond to the Research Questions defined in Chapter 1. They all compared personalized strategies with baseline methods from state-of-the-art literature to validate the premises of this research.

Experiment 1 was called Literals Similarity (LiSim) and computed the similarity between two resources in a LOD graph, considering the literal properties instead of the link properties. We compared the LiSim measure to pure-link approaches using two baseline methods, LDSD and ReSim. We concluded that the information stored in RDF literals enhanced the recommendations.

Experiment 2 was called Personalized Linked Data Semantic Distance (PLDSD) because it defines a generic method that personalizes the user model by weighing properties according to the user's past choices and then combines the personalized user model with the baseline LDSD measure. Results showed that the personalization approach increased the accuracy of recommendations in the context of the experiment.

Experiment 3 was called Summarization, and its goal was to validate the hypothesis that was filtering the most domain-relevant links to more effective recommendations in Linked Data-based systems. To accomplish this, we implemented a feature selection method based on statistics that filters the K most relevant properties before submitting the dataset to the personalization step (PLDSD method). The evaluation results show the best values for PLDSD combined with a $k = 10$ choice of feature selection strategy, outperforming the unweighted and unfiltered baseline method LDSD.

The results from the three experiments indicate that the specific goals defined in Chapter 1 were successfully achieved (see Chapter 6 for further details).

7.2 LIMITATIONS AND POINTS OF IMPROVEMENT

This section communicates the challenges involved in the research and ways for future developments. It explains how the limitations may affect the accuracy or applicability of the work and outlines the opportunities for future studies.

One limitation of this work is the size of the databases used to run the experiments. To perform a more comprehensive evaluation, it is suggested to use broader datasets. Even though the DBpedia dataset provides more than 4 million resources and more than 3 billion RDF triples, building high-quality training sets is not easy. For this work, we manually executed this task using exploratory SPARQL queries and made decisions based on common sense. This is considered a good strategy, although it leads to a reduced dataset.

In Experiment 1, despite the promising results, several precautions must be taken regarding literals containing more than one value. In the current development, we consider only the first value retrieved, which limits our work, as the other values are missed. Another issue is the pair of literals that may denote the same value. For instance, both <15^^xsd:byte> and <15.0^^xsd:decimal> denote the same value, fifteen. Although the intended meaning of the underlying literals is fifteen, our approach does not compute them because the data types are different.

Because of the inherently open nature of LOD, values stored in literals may not be properly formatted. Therefore, filtering out invalid values is a major concern when dealing with this type of information. For instance, in the following RDF statement from Figure 2.6: <dbr:Sao_Paulo,dbo:foundingDate,25-01-1554^^xsd:date>, the object value could also be found as "16th centur" of type String, which would be troublesome for someone interested in a quantifiable value of that particular property. However, this issue is not addressed in our approach. All these observed issues will be revised and resolved in future work.

Regarding Experiment 2, one great advantage of our personalization model is that it is generic enough to be applied to any LOD-based database. Although, we did not test it against a domain other than the movie domain. Experiments using the Personalized Linked Data Semantic Distance (PLDSD) with the music dataset is a current work-in-progress. Another limitation reported during the experimentation phase is about using the personalization method together with different semantic similarity measures; for each new measure to be aggregated to the model, we need to study a new function and find the best way to insert the $W(p_j, u_k)$ function in it. However, we plan future work to do at least one more experiment using other similarity measures than the LDSD.

Finally, regarding Experiment 3, one difficulty was to define the number of properties that should be used in the pre-selection process, that is, to define the value of k. Due to this, the tests were performed considering three distinct values for k. We defined the values so that the algorithm could explore a low ($k = 10$), a medium ($k = 100$), and a high ($k = max$) value of k for feature selection. The highest value of k is the maximum number of properties returned by the pre-selection step with the dataset used in the tests, which is $k = 535$ for movies and $k = 512$ for music. Although most of the results were more relevant when $k = 10$, both for LDSD and PLDSD, it is necessary to establish a method that identifies the optimal amount of properties for each data set.

7.3 FUTURE WORK

In future work, we aim to resolve the aforementioned limitations, which are:

- Solve the problems concerning invalid values and different data types in numeric typed literals of the LiSim method. For instance, the two values: <5.0^^(xsd: double)> and <5^^(xsd:integer)> are interpreted as different values, although they are the same in semantic terms;

- Perform new experiments integrating the LiSim method with the Personalized Linked Data Semantic Distance (PLDSD) method to incorporate as many resource features as possible to the recommendation algorithm. At this point in the research, the personalization step only integrates with similarity measures based on links, and does not take advantage of the semantics of literals in the user model personalization step;

- Include other LOD-based similarity measures in the personalization step to compare with PLDSD and determine which of them is best performing. The architecture of the solution presented in chapter 5 allows the addition of new semantic similarity or feature selection algorithms at specific points in the implementation, as long as the rules written in chapter 5 are followed.

- Conduct further studies on the feature selection task by comparing other LOD-driven approaches to the baseline methods used (NOIA et al., 2018; SILVA; DURãO; CAPRETZ, 2019);
- Regarding k optimal value issue, one suggestion is to apply the Elbow Curve, a popular method for finding the optimal number of clusters when working with the K-Means classification algorithm (KAUFMAN; ROUSSEEUW, 1990). We propose to investigate the possibility of finding the ideal value to K by adapting the Elbow Method to improve the results from this work.

Besides the suggestions from research limitations, we also identified opportunities for expanding this work by incorporating state-of-the-art research fields in LOD-based Recommender Systems. One possible future work is to evaluate the whole model of personalization strategies proposed by this thesis using a cross-domain dataset. This will enable the development of multi-domain recommendations for general use in linked datasets with a personalization approach. For instance, instead of recommending movies based only on past-rated movies, the system can recommend movies by analyzing songs or books rated by the user.

Regarding the pre-steps of dataset cleaning and enriching, we suggest developing semi-automatic methods to extract and gather DBpedia resources to enhance data semantics. In the present stage of work, we combine two previously mapped datasets available on the web with our raw data. We also undertake manual cleaning and ensure that links are not duplicated or missing. In addition, we plan to improve our cached query methods so that the algorithms have less access to the SPARQL endpoints on the Web, thus reducing the computational resources needed to process such a large amount of data.

The concern regarding exploiting hierarchy emerged during our research, as mentioned in Chapter 4. This sounds like an interesting approach to enhance our similarity models in future work. We suggest a method that exploits the entire entity hierarchy and selects the most relevant classes for retrieving and integrating links into the user model. In this way, it will be possible to calculate the semantic distance between resources considering all the semantic characteristics from their class hierarchy.

Another promising research field that could be incorporated into our work is RS explanations. A recommender system that provides explanations for its recommendations can benefit in the following ways: i) Increase user trust; ii) Improve user satisfaction; iii) Better system transparency and accountability; iv) Potential for user feedback; and v) Enhance domain knowledge acquisition by learning from users' feedback and preferences expressed through explanations.

Finally, we have made the complete project available under the Github URL: <https://github.com/gbrlamota/lodweb-pldsd>. Researchers interested in replicating and conducting further comparative studies can find the necessary documentation at this location. We also intend to make the data and coding available as an Application Programming Interface (API) for facilitating further extension and reuse.

7.4 DISSEMINATION

The D.Sc. study described in this thesis originated from the publications listed below.

- DA SILVA, GABRIELA ; DO NASCIMENTO, LARA ; DURAO, FREDERICO. Exploiting Linked Data-based Personalization Strategies for Recommender Systems. In: 18th International Conference on Web Information Systems and Technologies, 2022, Valletta. Proceedings of the 18th International Conference on Web Information Systems and Technologies, 2022. p. 226.
- SILVA, GABRIELA OLIVEIRA MOTA DA; SOUZA, PAULO ROBERTO DE ; DURAO, FREDERICO ARAUJO ; O, N.A. HSLD: a hybrid similarity measure for linked data resources. INTERNATIONAL JOURNAL OF METADATA, SEMANTICS, AND ONTOLOGIES (PRINT), v. 14, p. 16-25, 2020.
- DA SILVA, GABRIELA OLIVEIRA MOTA; CAPRETZ, MIRIAM; ARAUJO DURAO, FREDERICO. PLDSD: Personalized Linked Data Semantic Distance for LOD-Based Recommender Systems. In: The 21st International Conference on Information Integration and Web-Based Applications Services, 2019, Munich. New York: ACM, 2019. p. 294-303.
- RODRIGUES, MARIVALDO BISPO; DA SILVA, GABRIELA O. MOTA; DURAO, FREDERICO ARAUJO. User Models Development Based on Cross-Domain for Recommender Systems. In: the 22nd Brazilian Symposium, 2016, Teresina. Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web - Webmedia '16. New York: ACM Press, 2016. p. 363.
- DA SILVA, G. O. M.; CUNHA, M. V. ; DURAO, FREDERICO ARAUJO ; PEREIRA, H. B. B. . RDFREE: Um modelo computacional para analisar redes a partir de dados da Web Semântica. In: XIX Encontro Nacional de Modelagem Computacional e VII Encontro de Ciência e Tecnologia de Materiais, 2016, João Pessoa - PB. Anais do XIX Encontro Nacional de Modelagem Computacional e VII Encontro de Ciência e Tecnologia de Materiais, 2016. v. 1. p. 1-10.

7.5 SUMMARY

This chapter concludes the work by presenting the final remarks regarding general contributions (specific contributions of each experiment were presented in Chapter 6), points for improvement, and suggestions for future work.

BIBLIOGRAPHY

ADOMAVICIUS, G.; TUZHILIN; ALEXANDER. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, Institute of Electrical and Electronics Engineers (IEEE), v. 17, n. 6, p. 734–749, jun 2005.

AGGARWAL, C. C. *Recommender systems: the textbook.* [S.l.]: Springer International Publishing, 2016.

ANGLES, R.; GUTIERREZ, C. The expressive power of sparql. In: SPRINGER. *International Semantic Web Conference.* [S.l.], 2008. p. 114–129.

BARMAN, A.; TEWARI, A. S. Collaborative recommendation system using dynamic content based filtering, association rule mining and opinion mining. *International Journal of Intelligent Engineering and Systems*, The Intelligent Networks and Systems Society, v. 10, n. 5, p. 57–66, oct 2017.

BERNERS-LEE, T. Linked-data design issues. w3c design issue document. *The World-Wide Web Consortium W3C*, 2009.

BERNERS-LEE, T. Design issues: Linked data (2006). *URL http://www.w3.org/DesignIssues/LinkedData.html*, 2011.

BERNERS-LEE, T.; FIELDING, R.; MASINTER, L. Rfc 3986. *Uniform Resource Identifier (URI): Generic Syntax*, InternetEngineering Task Force, 2005.

BERNERS-LEE, T. et al. The semantic web. *Scientific american*, New York, NY, USA:, v. 284, n. 5, p. 28–37, 2001.

BIZER, C.; HEATH, T.; Berners-Lee, T. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, v. 5, n. 3, p. 1–22, 2009.

BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked data-the story so far. *Semantic services, interoperability and web applications: emerging concepts*, p. 205–227, 2009.

BRICKLEY, D.; MILLER, L. *FOAF Vocabulary Specification.* [S.l.], 2004. Http://xmlns.com/foaf/0.1/. Disponível em: <http://xmlns.com/foaf/0.1/>.

BURKE, R. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, Kluwer Academic Publishers, Hingham, MA, USA, v. 12, n. 4, p. 331–370, nov. 2002. ISSN 0924-1868.

CAO, Y. et al. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In: *The World Wide Web Conference*. New York, NY, USA: Association for Computing Machinery, 2019. (WWW '19), p. 151–161. ISBN 9781450366748. Disponível em: <https://doi.org/10.1145/3308558.3313705>.

CATHERINE, R.; COHEN, W. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In: *Proceedings of the 10th ACM Conference on Recommender Systems*. New York, NY, USA: ACM, 2016. (RecSys '16), p. 325–332. ISBN 978-1-4503-4035-9.

CHENIKI, N. et al. Lods: A linked open data based similarity measure. *2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, p. 229–234, 2016.

CODINA, V.; RICCI, F.; CECCARONI, L. *Exploiting the Semantic Similarity of Contextual Situations for Pre-filtering Recommendation*. [S.l.]: Springer, Berlin, Heidelberg, 2013.

CONTRIBUTORS, D. How to edit the dbpedia ontology. In: ______. *DBpedia Mappings*. DBpedia.org, 2022. Disponível em: <https://mappings.dbpedia.org/index.php/How_to_edit_the_DBpedia_Ontology/>. Acesso em: March 9th, 2022.

DAVOODI, E.; KIANMEHR, K.; AFSHARCHI, M. A semantic social network-based expert recommender system. *Applied Intelligence*, Springer Nature, v. 39, n. 1, p. 1–13, oct 2013. ISSN 1573-7497.

DIETZ, J. L. *What is Enterprise Ontology?* [S.l.]: Springer, 2006.

DU, Y. et al. Post-hoc recommendation explanations through an efficient exploitation of the dbpedia category hierarchy. *Knowledge-Based Systems*, v. 245, p. 108560, 2022. ISSN 0950-7051. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950705122002490>.

FERRÉ, S. Sparklis: a sparql endpoint explorer for expressive question answering. In: *ISWC posters & demonstrations track*. [S.l.: s.n.], 2014.

FIELDING, R. et al. *Hypertext transfer protocol–HTTP/1.1*. [S.l.], 1999.

FRESSATO, E. P. *Incorporação de metadados semânticos para recomendação no cenário de partida fria*. 105 p. Tese (Doutorado) — Universidade de São Paulo, 2019.

GAN, L. et al. Emdkg: Improving accuracy-diversity trade-off in recommendation with em-based model and knowledge graph embedding. In: *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. New York, NY, USA: Association for Computing Machinery, 2021. (WI-IAT '21), p. 17–24. ISBN 9781450391153. Disponível em: <https://doi.org/10.1145/3486622.3493925>.

GARCÍA, C. G. et al. Social recommender system: A recommender system based on tweets for points of interest. In: *Proceedings of the 4th Multidisciplinary International Social Networks Conference on ZZZ - MISNC '17*. New York, NY, USA: ACM Press, 2017. (MISNC '17), p. 28:1–28:7. ISBN 978-1-4503-4881-2.

GARSHOL, L. M. Living with topic maps and rdf. *Online only*, Citeseer, v. 13, 2003.

GEMMIS, M. de et al. Semantics-aware content-based recommender systems. In: RICCI, F.; ROKACH, L.; SHAPIRA, B. (Ed.). *Recommender Systems Handbook*. Boston, MA: Springer US, 2015. p. 119–159. ISBN 978-1-4899-7637-6.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, Elsevier, v. 43, n. 5-6, p. 907–928, 1995.

GUHA, R.; BRICKLEY, D. W3C Recommendation, *RDF Schema 1.1*. 2014. Disponível em: <http://www.w3.org/TR/2014/REC-rdf-schema-20140225/>.

GUO, G. Resolving data sparsity and cold start in recommender systems. In: MASTHOFF, J. et al. (Ed.). *User Modeling, Adaptation, and Personalization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 361–364. ISBN 978-3-642-31454-4.

GUO, S.; ALAMUDUN, F.; HAMMOND, T. Résumatcher: A personalized résumé-job matching system. *Expert Systems with Applications*, Elsevier, v. 60, p. 169–182, 2016.

GUY, I. People recommendation on social media. In: _____. *Social Information Access: Systems and Technologies*. Cham: Springer International Publishing, 2018. p. 570–623. ISBN 978-3-319-90092-6.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, JMLR.org, v. 3, p. 1157–1182, mar. 2003. ISSN 1532-4435.

HERLOCKER, J. L. et al. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, Association for Computing Machinery (ACM), v. 22, n. 1, p. 5–53, jan 2004.

HOLZE, J. Dbpedia snapshot 2022-12 release. In: _____. *DBpedia Archive: Announcement*. DBpedia.org, 2023. Disponível em: <https://www.dbpedia.org/blog/dbpedia-snapshot-2022-12-release/>. Acesso em: March 27th, 2023.

HUTT, K. A comparison of rdf query languages. In: *Proc. of 21th Computer Science Seminar, Hartfort, Connecticut*. [S.l.: s.n.], 2005. p. 1–7.

ISINKAYE, F.; FOLAJIMI, Y.; OJOKOH, B. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, Elsevier BV, v. 16, n. 3, p. 261–273, nov 2015.

JäRVELIN, K.; KEKäLäINEN, J. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, ACM, New York, NY, USA, v. 20, n. 4, p. 422–446, out. 2002. ISSN 1046-8188.

JAWAHEER, G.; WELLER, P.; KOSTKOVA, P. Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. *ACM Transactions on Interactive Intelligent Systems*, Association for Computing Machinery (ACM), v. 4, n. 2, p. 1–26, jun 2014.

JOSEPH, K.; JIANG, H. Content based news recommendation via shortest entity distance over knowledge graphs. In: *Companion Proceedings of The 2019 World Wide Web Conference*. New York, NY, USA: ACM, 2019. (WWW '19), p. 690–699. ISBN 978-1-4503-6675-5.

KAUFMAN, L.; ROUSSEEUW, P. J. *Finding Groups in Data: An Introduction to Cluster Analysis.* [S.l.]: John Wiley, 1990. ISBN 978-0-47031680-1.

KIM, M. H. J. G. 5(star) open data. In: _____. *5stardata.info*. Content freely available under the CC0 Public Domain Dedication, 2019. Disponível em: <https://5stardata.info/en/>. Acesso em: June 6th, 2019.

KLUVER, D.; EKSTRAND, M. D.; KONSTAN, J. A. Rating-based collaborative filtering: Algorithms and evaluation. In: _____. *Social Information Access.* [S.l.]: Springer International Publishing, 2018. cap. 10, p. 344–390. ISBN 978-3-319-90092-6.

KLYNE, G.; CARROLL, J. J. Resource description framework (rdf): Concepts and abstract syntax. 2006.

KOBILAROV, G. et al. Media meets semantic web - how the bbc uses dbpedia and linked data to make connections. In: *ESWC.* [S.l.: s.n.], 2009.

KUMAR, P.; THAKUR, R. S. Recommendation system techniques and related issues: a survey. *International Journal of Information Technology*, Springer Nature, v. 10, n. 4, p. 495–501, apr 2018.

LAM, X. N. et al. Addressing cold-start problem in recommendation systems. In: *Proceedings of the 2Nd International Conference on Ubiquitous Information Management and Communication.* New York, NY, USA: ACM, 2008. (ICUIMC '08), p. 208–211. ISBN 978-1-59593-993-7.

LEHMANN, J. et al. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, v. 6, n. 2, p. 167–195, 2015. Disponível em: <http://jens-lehmann.org/files/2015/swj_dbpedia.pdf>.

LÜ, L. et al. Recommender systems. *Physics Reports*, Elsevier BV, v. 519, n. 1, p. 1–49, oct 2012.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to Information Retrieval.* New York, NY, USA: Cambridge University Press, 2008. ISBN 0521865719, 9780521865715.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. Evaluation in information retrieval. In: _____. *Introduction to Information Retrieval.* [S.l.]: Cambridge University Press, 2008. p. 139–161.

MCCRAE, J. P. The linked open data cloud. In: _____. *The Linked Open Data Cloud.* Insight Centre for Data Analytics, 2023. Disponível em: <https://lod-cloud.net/>. Acesso em: September 3rd, 2023.

MUSTO, C. et al. Semantics-aware graph-based recommender systems exploiting linked open data. In: *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization.* New York, NY, USA: ACM, 2016. (UMAP '16), p. 229–237. ISBN 978-1-4503-4368-8.

NATARAJAN, S. et al. Cd-semmf: Cross-domain semantic relatedness based matrix factorization model enabled with linked open data for user cold start issue. *IEEE Access*, v. 10, p. 52955–52970, 2022. Disponível em: <https://doi.org/10.1109/ACCESS.2022.3175566>.

NOIA, T. D. et al. Using ontology-based data summarization to develop semantics-aware recommender systems. In: GANGEMI, A. et al. (Ed.). *The Semantic Web.* Cham: Springer International Publishing, 2018. p. 128–144. ISBN 978-3-319-93417-4.

NOIA, T. D. et al. Linked open data to support content-based recommender systems. In: *Proceedings of the 8th International Conference on Semantic Systems.* New York, NY, USA: ACM, 2012. (I-SEMANTICS '12), p. 1–8. ISBN 978-1-4503-1112-0.

PAN, J. Z. Resource description framework. In: *Handbook on ontologies.* [S.l.]: Springer, 2009. p. 71–90.

PARRA, D.; SAHEBI, S. Recommender systems: Sources of knowledge and evaluation metrics. In: _____. *Advanced Techniques in Web Intelligence-2: Web User Browsing Behaviour and Preference Analysis.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 149–175. ISBN 978-3-642-33326-2.

PASSANT, A. Measuring semantic distance on linking data and using it for resources recommendations. In: *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence.* [S.l.]: AAAI, 2010.

PÉREZ, J.; ARENAS, M.; GUTIERREZ, C. Semantics and complexity of sparql. *ACM Transactions on Database Systems (TODS)*, ACM, v. 34, n. 3, p. 16, 2009.

PERRY, M.; HERRING, J. Ogc geosparql-a geographic query language for rdf data. *OGC Implementation Standard. Sept*, 2012.

PIAO, G.; ARA, S. s.; BRESLIN, J. G. Computing the semantic similarity of resources in dbpedia for recommendation purposes. In: QI, G. et al. (Ed.). *Semantic Technology*. Cham: Springer International Publishing, 2016. p. 185–200.

PIAO, G.; BRESLIN, J. G. Measuring semantic distance for linked open data-enabled recommender systems. In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2016. (SAC '16), p. 315–320. ISBN 978-1-4503-3739-7.

PIAO, G.; BRESLIN, J. G. Inferring user interests in microblogging social networks: a survey. *User Modeling and User-Adapted Interaction*, Springer Nature America, Inc, v. 28, n. 3, p. 277–329, aug 2018.

REUSENS, M. et al. A note on explicit versus implicit information for job recommendation. *Decision Support Systems*, Elsevier BV, v. 98, p. 26–35, jun 2017.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Recommender systems: introduction and challenges. In: *Recommender systems handbook*. [S.l.]: Springer International Publishing, 2015. p. 1–34.

SARKER, M. K. et al. Explaining trained neural networks with semantic web technologies: First steps. *Proceedings of the Twelveth International Workshop on Neural-Symbolic Learning and Reasoning*, 2017.

SARWAR, B. et al. Item-based collaborative filtering recommendation algorithms. In: *Proceedings of the tenth international conference on World Wide Web - WWW '01*. [S.l.]: ACM Press, 2001. (WWW '01), p. 285–295. ISBN 1-58113-348-0.

SAVESKI, M.; MANTRACH, A. Item cold-start recommendations: learning local collective embeddings. In: *Proceedings of the 8th ACM Conference on Recommender systems - RecSys '14*. [S.l.]: ACM Press, 2014. p. 89–96.

SEDHAIN, S. et al. Social collaborative filtering for cold-start recommendations. In: *Proceedings of the 8th ACM Conference on Recommender Systems*. New York, NY, USA: ACM, 2014. (RecSys '14), p. 345–348. ISBN 978-1-4503-2668-1.

SHANI, G.; GUNAWARDANA, A. Evaluating recommendation systems. In: ______. *Recommender Systems Handbook*. Boston, MA: Springer US, 2011. p. 257–297. ISBN 978-0-387-85820-3.

SHI, Y. et al. Climf: Learning to maximize reciprocal rank with collaborative less-is-more filtering. In: *Proceedings of the sixth ACM conference on Recommender systems - RecSys '12*. New York, NY, USA: ACM Press, 2012. (RecSys '12), p. 139–146.

SILVA, G. O. M. da; DURãO, F. A.; CAPRETZ, M. Pldsd: Personalized linked data semantic distance for lod-based recommender systems. In: *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications and Services*. New York, NY, USA: Association for Computing Machinery, 2019. (iiWAS2019),

p. 294–303. ISBN 9781450371797. Disponível em: <https://doi.org/10.1145/3366030.3366041>.

TERÁN, L.; MENSAH, A. O.; ESTORELLI, A. A literature review for recommender systems techniques used in microblogs. *Expert Systems with Applications*, Elsevier BV, v. 103, p. 63–73, aug 2018.

THORAT, P. B.; GOUDAR, R. M.; BARVE, S. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, Foundation of Computer Science, v. 110, n. 4, p. 31–36, jan 2015.

TRIPERINA, E. et al. Creating the context for exploiting linked open data in multidimensional academic ranking. *International Journal of Recent Contributions from Engineering, Science & IT (iJES)*, v. 3, n. 3, p. 33–43, 2015.

ZHANG, F. et al. Alleviating the data sparsity problem of recommender systems by clustering nodes in bipartite networks. *Expert Systems with Applications*, v. 149, p. 113346, 2020. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417420301718>.

ZHANG, Z.-K. et al. Solving the cold-start problem in recommender systems with social tags. *EPL (Europhysics Letters)*, IOP Publishing, v. 92, n. 2, p. 28002, oct 2010.