Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Software Department

Improving Linked Open Data-based Recommendation System for Multi Domain

A Dissertation

Submitted to the Council of the College of Information Technology at the University of Babylon in Partial Fulfillment of the Requirements for the Doctor of Philosophy Degree in Information Technology / Software

By

Ahmed Mounaf Mahdi Mohammed

SUPERVISED BY

Prof. Dr. Asaad Sabah Hadi Abbas

2022 A.D. **1444 A.H.**

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

﴿ يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ ۚ وَاللَّهُ بِمَا تَعْمَلُونَ خَبِيرٌ ﴾

صَدَق الله العَلي العَظيم

سورة المجادلة – الآية (11)

Supervisor Certification

I certify that the dissertation entitled "Improving Linked Open Data-based Recommendation System for multi Domain" was prepared under my supervision at the department of Software/ College of Information Technology/ University of Babylon as partial fulfillment of the requirements of the degree of Doctor of Philosophy in Information Technology-Software.

Signature:

Supervisor Name: **Prof. Dr. Asaad Sabah Hadi Abbas**

Date: / /2022

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled "Improving Linked Open Data-based Recommendation System for multi domain" for debate by the examination committee.

Signature:

Name: **Assistant Prof. Dr. Ahmed Saleem Abbas**

Head of Software Department

Date: / /2022

Certification of the Examination Committee

We, the undersigned, certify that (Ahmed Mounaf Mahdi Al-musawi) candidate for the degree of Doctor of Philosophy in Information Technology-Software, has presented his dissertation of the following title (Improving Linked Open Data-based Recommendation System for multi Domain) as it appears on the title page and front cover of the dissertation that the said dissertation is acceptable in form and content and displays a satisfactory knowledge of the field of study as demonstrated by the candidate through an oral examination held on: Sep 28, 2022.

Signature:
Name: Dr. Ahmed Tariq Sadiq
Title: Professor
Date: / / 2022
(Chairman)

Signature:
Name: Dr. Huda Naji Nawaf
Title: Professor
Date: / / 2022
(Member)

Signature:
Name: Dr. Khalid Ali Hussein
Title: Professor
Date: / / 2022
(Member)

Signature:
Name: Dr. Ali Hadi Hasan
Title: Asst. Professor
Date: / / 2022
(Member)

Signature:
Name: Dr. Ahmed Saleem Abbas
Title: Asst. Professor
Date: / / 2022
(Member)

Signature:
Name: Dr. Asaad Sabah Hadi
Title: Professor
Date: / / 2022
(Member / Supervisor)

Signature:
Name: Prof. Dr. Hussein Atiya Lafta
Title: Professor
Date: / / 2022
(Dean of Collage of Information Technology)

Dedication

In the name of Allah, Most Greatest, Most Gracious, Most Merciful

All praise is for Allah who provides me the strength, inspiration and courage to complete this work.

Gratitude to you is not enough…

To my mother, without her prayers and her consent. I cannot succeed. I pray to God to protect her.

To my father, who always encourages me to complete my studies, and stand by me in every moment of my life.

To my dear wife, who had a great role in bearing the burden of responsibilities to devote my efforts for studying, a gratitude to her for her tremendous efforts in rearing and teaching my children until my daughter became one of the outstanding achievers. Thanks for always standing by our side.

To my brother and sisters for their continuous support.

Last but not least, thanks for everybody who contributed in one way or another to accomplish this work.

Acknowledgements

A special thanks to my supervisor Prof. Dr. Asaad Sabah Hadi for his supervision and guidance throughout my dissertation. There are no words would express my appreciation for his valuable comments and suggestions. I would also like to thank all professors who have been great contributors in building up my experience and knowledge.

Abstract

The Semantic Web and Linked Open Data (LOD) make the data readable by machines as well as users. LOD, a set of principles for publishing and linking structured data on the web, opens up opportunities for researchers to utilize the huge amount of data available to develop applications in different domains. Recommendation Systems (RSs) that depend on the ratings to recommend items suffer from the problem of cold-start, while others that depend on the content suffer from lack of semantic and limited content analysis. Therefore, RS needs to enrich items and users with extra features, by exploiting the knowledge encoded in LOD to solve the common problems of traditional RSs, and calculates the semantic distance between the LOD resources in order to produce recommendations.

Most previous studies depend on pure link-based similarity measures, which will degrade the accuracy of recommendation if there are insufficient links between the resources even though they share many features. In addition, the well-known linked data semantic distance (LDSD) depends on the number of direct and indirect links connecting resources and treating them equally. In this dissertation, an Enhanced LDSD (ELDSD) measure is proposed to compute resource similarity based on direct and indirect links and by increasing the impact of having direct links at the expense of indirect links. In addition, to combine it with a measure that depends on the features extracted from two datasets after representing them using Word2Vec model to recommend relevant movies. In addition to recommend items from other domains related to the recommended movies such as related music and book.

The experiment evaluation utilizing the MovieLens and Yahoo Movie datasets are carried out and the results indicate that the proposed approach outperforms other approaches that depend on links structure with up to 7.8% improvement in f-measure and with up to 31.8% improvement over other approaches that depend on the features only.

Publications

- A. M. Mahdi and A. S. Hadi, "Utilizing LOD Relationships and FOAF Vocabularies for Top-N Recommender System," *2021 1st Babylon International Conference on Information Technology and Science (BICITS)*, 2021, pp. 98-103, doi: 10.1109/BICITS51482.2021.9509914. **(Published)**
- A. M. Mahdi and A. S. Hadi, "The Current State of Linked Data-based Recommender Systems," *2021 2nd Information Technology To Enhance e-learning and Other Application (IT-ELA)*, 2021, pp. 154-160, doi: 10.1109/IT-ELA52201.2021.9773738. **(Published)**
- Linked Data-based Recommender System Using Hybrid Semantic Similarity Measure, *International Conference on Scientific Research and Innovation (ICSRI 2022)* April 11-12, 2022, in Cincinnati, Ohio, USA. **(Presented)**

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Meaning
ABSTAT	ABstraction and STATistics
BNB	British National Bibliography
CBF	Content-based Filtering
CBOW	Continuous Bag of Words model
CF	Collaborative Filtering
DBpedia	Database pedia
DCMI terms	Dublin Core Metadata Initiative terms
ELDSD	Enhanced LDSD
EPICS	Extended Partitioned Information Content Similarity
FOAF	Friend of a Friend
GGG	Giant Global Graph
IRIs	Internationalized Resource Identifiers
LD	Linked Data
LDA	Latent Dirichlet Allocation
LDSD	Linked Data Semantic Distance
LinkedMDB	Linked Movie Database
LOD	Linked Open Data
LSA	Latent Semantic Analysis
MRR	Mean Reciprocal Rank
OGD	Open Government Data
OWL	Web Ontology Language
PIC	Partitioned Information Content
PICSS	Partitioned Information Content Semantic Similarity
PLDSD	Propagated Linked Data Semantic Distance

RDF	Resource Description Framework
RDFS	RDF Schema
RQ	Research Question
RS	Recommendation System
RSs	Recommendation Systems
SW	Semantic Web
SKOS	Simple Knowledge Organization System
SPARQL	Simple Protocol and RDF Query Language
SVD	Singular Value Decomposition
TLDSD	Typeless Linked Data Semantic Distance
TSV	Tab Separated Values
URIs	Uniform Resource Identifiers
URLs	Uniform Resource Locators
W3C	The World Wide Web Consortium
wPLDSD	Weighted Propagated Linked Data Semantic Distance
wTLDSD	Weighted Typeless Linked Data Semantic Distance

CHAPTER ONE

GENERAL INTRODUCTION

1.1 Introduction

The traditional Web is a web of text and pictures, which is helpful for users, but computers play a very small role on this Web [1]. Tim Berners-Lee recognized the importance of incorporating semantics into the Web to publish structured data and extend its functionality, and in 1994, the Semantic Web was born. The publishing of structured data on the Web and making links between them by depending on some principles is known as Linked Data (LD), which provides the datasets as Resource Description Framework (RDF) model to achieve the Semantic Web (SW) [2]. RDF is a graph-based data model contains a collection of triples (subject (s), predicate (p), and object (o)) which represent directed labelled edges $s \xrightarrow{p} o$ in a graph. RDF has gained significant adoption in the past years, particularly on the Web[3].

Large amounts of RDF data has been published in publicly accessible datasets linked together to create the Linked Open Data (LOD) cloud. RDF data is widely available on the Web of Data, but only a small number of applications use their full potential [4]. LOD is used in many applications, and a Recommendation System (RS) is one that uses LOD extensively and has benefited from its use [5]. RS is a critical tool for analyzing data and automatically providing suggestions to users. At the moment, practically all applications employ RS to deliver more accurate suggestions to users in order to entice users to use their applications for an extended period of time. These apps touch practically every aspect of people's life[6]. Information encoded in the LOD cloud helps to solve problems where little or no features can characterize the items to be suggested. Many features relevant to a recommendation task may be retrieved from the LOD cloud to enhance the quality of recommended items and satisfy the user’s needs[4, 7].

In this dissertation, a linked data-based RS for multi-domain is designed that depends on linked data by utilizing the transversal relations existing in the Linked

Movie Database (LinkedMDB) and Database pedia (DBpedia) datasets as well as the hierarchical structure in DBpedia to find the candidate resources for the recommendations in movie domain and to recommend the related book and the featured song of the recommended movies.

There are some research questions, which will be answered throughout this dissertation

RQ1. How to fix the missing links between LinkedMDB and DBpedia in order to fetch the features of the movies from both datasets? (Answered in Section 3.4 P56)

RQ2. How to fix the links to deprecated dataset such as Mashup Book that links a movie to related book and How to recommend items from domains other than movies such as books or music? (Answered in Section 3.4 P59)

RQ3. How to represent the properly formatted item's features that extracted from two datasets in a unified method to facilitate the production of accurate recommendations? (Answered in Section 3.6 P63) and How to represent the user's profile? (Answered in Section 3.7 P64)

RQ4. How the different kinds of relations exist in LOD cloud be utilized to produce recommendations? (Answered in Section 3.8.1 P69)

RQ5. How to rank the candidate items and find the best similarity measure to rank and produce the Top-N recommendations? (Answered in Section 3.8.2 P80)

1.2 Related works

The previous works contributed to the linked data-based recommender systems in different ways; proposing or extending a similarity measure, a new algorithm for utilizing LOD and representing the features, information gathering

and enrichment of users and items, and building a new ontology for specific domain.

1.2.1 Proposing or extending a similarity measure

There are several methods for estimating the similarity of the resources to recommend related items: link-based, feature-based, statistical, and hybrid methods. Link-based methods such as Linked Data Semantic Distance (LDSD) [8] in 2010, Propagated Linked Data Semantic Distance (PLDSD) [9] in 2017, and Typeless Linked Data Semantic Distance (TLDSD) [10] in 2019 depend on the graph structure of LOD to find the similarity between two resources. Feature-based methods such as Partitioned Information Content Semantic Similarity (PICSS)[11] in 2016 and Extended Partitioned Information Content Similarity (EPICS) [12] in 2020 estimate the similarity of the resources by determining the similarity of their feature sets. Statistical-based similarity methods depend on statistical data obtained from resources, whereas hybrid methods combine several of these methods into a single approach [10].

LDSD assumes that a high number of direct and indirect links between the resources leads to high relatedness. In this measure, the direct and indirect links are treated equally. Variations of LDSD, namely PLDSD and its weighted version (wPLDSD) was proposed in 2017 to consider the indirect links that have more than one intermediate node [9, 10]. Another measure called TLDSD and its weighted version (wTLDSD) are proposed in 2019 by [10] to deal with cases when multiple links have varying properties in LOD.

Meymandpour and Davis [11] in 2016 proposed a measure called PICSS which depends on Partitioned Information Content (PIC) to calculate the information content of the resources using three sets of features: the common features of the two resources, the distinct features of the first resource and the

distinct features of the second resource. Komeiha *et al.* [12] in 2020 proposed a variation of PICSS called EPICS which uses the same set of features used in PICSS and adds a set of similar features of the two resources.

Boubenia, *et al.* [13] in 2020 investigated the possibility of integrating similarity and relatedness in LOD and proposed a new LOD-based similarity metric which can be beneficial in Online Social Networks (OSN) and specifically in cross-OSN recommendation. The authors demonstrated that focusing solely on the principle of relatedness is heavily influenced by the sparsity of relations and does not always expose the true similarity between concepts. Insufficient content of resources, on the other hand, has a negative effect on finding the true similarity of LOD resources when depending solely on the similarity of the resources content. Furthermore, the authors showed during their extensive review that evaluating the concepts in terms of common properties could be less reflective of the true similarities. They assumed that comparing the values of these properties would make more sense and strengthen the relation, and more recommendations would be gained by comparing the links as well as the content of LOD concepts. When similarities and relatedness are combined, the findings indicate a significant improvement in terms of accuracy.

Silva, *et al*. [14] in 2020 proposed a similar idea of a hybrid similarity measure that takes into account information included in RDF literals in addition to the resources' links. The suggested method was evaluated using data from DBpedia in the scope of a LOD-based RS. The findings of the experiments show that the combination of the LDSD measure with the semantic distance measure between the literals of two items increases the quality of the provided recommendations. However, literals may not be properly formatted or have invalid values that affect the quality of the selected features.

Yadav, *et al.* [15] in 2020 proposed a technique based on an ontology for determining semantic similarity between two items, and another technique based on user rating, as well as a hybrid of the two techniques. Item-based semantic similarity is calculated using the binary Jaccard similarity measure. The similarity between the two items is computed for each attribute in the ontology. In this case, all of the characteristics related to the items' classes and subclasses will be examined as well as the users' explicit ratings to establish the overall item semantic similarity score.

Sebbaq, *et al.* [16] in 2020 suggested to use the semantic web, LOD, ontologies, and their mappings as a basis in their approach. Their solution addressed the cold-start problem by using semantic similarity algorithms and Linked Data to increase the efficiency of content-based RS. They also used the semantic web and ontologies to describe domain knowledge and model user interests. In LOD, the syntactic and semantic similarity between items and properties is assessed. They utilized the Dice Index as a term-based measure, Cosine Similarity as a vector-based measure, and WordNet as a semantic-based measure. They discovered that by combining the various methods described, they were able to overcome the limitations of conventional RS.

1.2.2 Information gathering and enrichment

Linked data is needed in various kinds of applications for enriching some concepts or items used in the application. One of the most popular application uses linked data for this purpose is RS. Therefore, researchers in this field of study use different datasets to gather extra information from LOD for enrichment of the items and users.

Musto, *et al.* [17] in 2017 utilized the LOD for both recommendation and explanation jobs. They represented the items using two types of features; the first

is structured data derived from the LOD cloud, referred to as LOD-based features, such as a movie's genre or the author of a book. The second is the graph-based features which are dependent on the bipartite and tripartite graph-based representations' topological properties. The authors implemented three different classification algorithms: Random Forests, Naive Bayes, and Logistic Regression and they conducted several experiments and found that using LOD can improve the accuracy of the RS.

Iana, *et al.* [18] in 2019 presented a method for recommending a list of conferences to assist authors in selecting those conferences that are relevant to their articles. SciGraph was used to collect data on previously published works and prior conferences, while WikiCFP was used to collect data about future conferences. SciGraph is a LOD dataset released by Springer Nature, while WikiCFP is a website for the upcoming conference that includes deadlines and other information needed for recommending related conferences. Therefore, they developed a crawler for WikiCFP to gather information about future conferences. The author's name, abstract, or keywords were used to make the recommendation. The authors employed similarity approaches and machine learning techniques and found that recommendations based on keywords outperformed those based on abstracts, while those based on abstracts outperformed those based on authors.

Natarajan, *et al.* [19] in 2020 presented a method to overcoming two of the most prevalent difficulties in collaborative filtering RS: cold start and data sparsity. They developed a linked data-based RS to address the cold start issue and they used Matrix Factorization model with Linked Open Data for solving the data sparsity problem. They used the DBpedia dataset to gather more information about the newly created items in order to overcome the cold-start issue. The experimental assessment revealed that the suggested method outperformed the alternatives evaluated by the authors.

1.2.3 Proposing an Algorithm

Most approaches focus on proposing an algorithm to extract the features and exploit the knowledge exist in LOD in a specific way and to combine a set of methods to extract, represent, and utilize the different kinds of features available in different datasets and in different domains.

Oliveira, *et al*. [20] in 2017 suggested three types of recommendations: the first type recommends a list of resources that share the same input's type. The second recommends individuals who are in the same domain as the initial resource. The third will recommend resources of different kinds that are related to the initial resource based on their textual descriptions. For the purposes of conducting tests and evaluating RS, a user study implemented by making a small group of specialists evaluate the system, and the feedback received was utilized to determine the accuracy, novelty, and utility of the proposed approaches.

Vagliano [21] in 2017 presented an algorithm based on Linked Data that recommends similar items based on the relationships between resources. The resource categories were examined and the direct connections to other resources were discovered. The approach makes use of the relationships contained in a single dataset, namely the DBpedia dataset. The experimental assessment is based on a user study and compares the RS's performance to that of other state-of-the-art algorithms. They discovered that including various types of relationships from LOD improved the prediction accuracy.

1.2.4 Building an Ontology and Other Purposes

Some research's contributions were to build an ontology for specific domain using data from LOD and others focus on applying some techniques beside linked data to enhance the accuracy of the recommendation.

Di Noia *et al*. [22] in 2018 developed a content-based RS that calculates the Jaccard similarity between the item **i** and item **j** in order to suggest the most related items to the user. They focused on the problem of feature selection and dimensionality in LOD datasets and on how to extract the most relevant features from LOD. In addition, The authors investigated the application of ontology-based data summarization for feature selection in LOD data for the purpose of recommending items from the movie, book, and music domains. They compared the findings of traditional feature selection techniques to those of Knowledge Graph Profiling with ABstraction and STATistics (ABSTAT), a schema summarizing tool, and discovered that the latter produces more accurate predictions in terms of precision and diversity. They also found that the features, such as: starring/actors, director and subject/genre, are the most useful features in movie domain.

Selvan, et al. [23] in 2019 suggested a fuzzy ontology-based method. Their method made use of the Type-2 fuzzy logic to recommend food and medicines in IoT-based healthcare systems. The LOD cloud is exploited to enhance patient records by creating connections between them and the corresponding LOD resources. The purpose of developing a fuzzy ontology is to automate the job of recommending meals and medicines to diabetes patients. LOD contains all available information about diseases. The method proposed was assessed using commonly used performance indicators, including precision, recall, f-measures, and accuracy.

Hsu and Lin [24] in 2020 explored how to connect governmental open data to Facebook profiles and how to recommend similar Open Government Data (OGD). Machine learning and semantic web technology integrated into a cloud computing framework. Following that, a linked data query implemented and linked to Facebook fan pages to identify highly relevant OGD based on recent

topics that users are reading on Facebook. They will also collect posts liked by users on Facebook and conduct a correlation analysis using K-nearest neighbors to classify each post. After that, an ontology is built, in addition to the rules to recognize each post.

In previous works, different datasets are used for gathering knowledge, and each dataset can be used for a specific domain. DBpedia is considered as a cross-domain dataset that is used intensively alone. Few studies have been conducted using DBpedia and other dataset for specific domain. The diversity of datasets in LOD opens up opportunities for the researchers to choose any dataset in different domains and contribute to this field of study.

Table 1.1. Summary of related works

Study	Authors	RS category	Contribution	Methods used	Dataset	Domain	Evaluation	Future works
[8]	**Passant 2010**	LOD-based RS	- LDSD	Depend on the number of direct and indirect links to find similarity	DBpedia	Music	1- Last.fm	1- Using the transitivity of genres 2- Link propagation 3- Feature selection 4- Using other LOD datasets
[11]	**Meymandpour and Davis 2016**	Hybrid RS	- Proposed PICSS measure	Depend on Partitioned Information Content (PIC)	DBpedia	Movies	2- MovieLens 100K and MovieLens 1M	5- None
[10]	**Alfarhood *et al.* 2017**	LOD-based RS	- Proposed PLDSD	Variation of LDSD considers indirect links that have more than one intermediate node	DBpedia	Music	- Dataset from the second Linked Open Data-enabled recommender systems challenge - F-measure and MRR	1- Combine this measure with other. 1- Analyzing the effects of propagating semantic distances on different domains.
[17]	**Musto *et al.* 2017**	Hybrid RS	- Proposed algorithm and enrichment	Random Forests Naïve Bayes Logistic Regression.	DBpedia	Movies Book Music	- MovieLens 1M DBBook Last.fm.	1- Representations of items with word embedding 2- Novelty & diversity

[20]	**Oliveira *et al*. 2017**	Content-based RS with LOD	- Proposed algorithm using two datasets to extract knowledge	SPARQL & string distance of values of properties	Dbpedia & LinkedMDB	Movies	- User study. Accuracy, Prediction accuracy by ranking, Novelty and Utility.	1- Apply to different domains 2- Take the user profile into account 2- Personalizing the output
[21]	**Vagliano 2017**	Content-based RS with LOD	- 1- Proposed algorithm. 2- New Ontology for user's context	LDSD	DBpedia	Movies	- User study	1- Apply to other domain. 2- Personalization. 3- Hybrid RS
[22]	**Di Noia *et al*. 2018**	Content-based RS with LOD	- Apply feature selection tools for enhancing RS	1- Jaccard sim of resources' features ontology-based data summarization for feature selection	DBpedia	Movies Books Music	- MovieLens, Last.FM, LibraryThing dataset Precision MRR Catalog coverage Aggrentropy	3- Use other parameters of ABSTAT to enhance the similarity measure
[9]	**Alfarhood *et al*. 2019**	LOD-based RS	- TLDSD	Variation of LDSD deals with cases when multiple links have varying properties in LOD	DBpedia	Music	- Dataset from the second Linked Open Data-enabled recommender systems challenge - F-measure, MRR and coverage	2- Include the textual content of resource properties in the similarity estimation. 3- Cross-domain recommendation

[18]	**Iana *et al*. 2019**	Content-based RS with LOD	- Enrichment	1- TF-IDF with n-gram 2- Similarity methods ML techniques	SciGraph WikiCFP	Research Field	- Publication data is used for evaluation Recall MAP	NONE
[23]	**Selvan *et al*. 2019**	Knowledge - based RS	- 1-Ontology building 2- Enrichment	3- Type-2 fuzzy logic	-Pubmed+ Medline plus -Fuzzy ontology created for patients and drug	Medicine	- Precision Recall F-measures Accuracy	1- Enhance patient health condition detection. 2- Incorporate neural networks and sentiment analysis into the treatment RS. Security mechanism for the medical data.
[12]	**Komeiha *et al*. 2020**	LOD-based RS	- EPICS - Proposed a library for implementing and testing of measures.	Variation of PICSS considers set of similar features	DBpedia	Cross-domain	- Wikipedia Similarity 353 benchmark with DBpedia dataset. - Pearson Correlation Coefficient.	1- Further develop LDS by taking into account more similarities. 2- Bring better scalability to LDS library
[13]	**Boubenia *et al*. 2020**	Content-based RS with LOD	Proposed similarity measure	Similarities and relatedness are combined	DBpedia	Cross-domain	1- DBpedia 2-Twitter 3- Youtube	3- Apply other datasets. 4- Weighting strategy for some features. 5- Apply fuzzy theory.

[14]	**Silva *et al.* 2020**	Content-based RS with LOD	Creating a Hybrid similarity measure	Similarities and Relatedness are combined	DBpedia	Movies and Music	Precision@K MAP NDCG	1- Test other link-based similarity measures. 2- Apply to other domains (books, games).
[15]	**Yadav *et al.* 2020**	Hybrid RS	Proposed Semantic Similarity	Jaccard based on ontology and explicit user rating	1- DBpedia 2- Facebook 4j API 3- Web crawler for IMDb	Movies	-MovieLens and Yahoo! Webscope R4 dataset. - MAE, P, R and F-measure	1- Automatic selection of user's feature. 2- Dimensional reduction 3- Novelty, Diversity
[16]	**Sebbaq *et al.* 2020**	Hybrid RS	Build an ontology for computer science	Syntactic and semantic similarity of resources	Data from: 1- Courser 2- Edx 3- Linkedin 4- Facebook then convert them to RDF	E-learning	For future work	1- Extra enhancement to the built ontology & enhance recommendation 2- Evaluation with other approaches
[19]	**Natarajan *et al.* 2020**	Collaborative-filtering with LOD	Proposed Similarity measure	1- Matrix Factorization with LOD. 2- Improvised Pearson correlation coefficient + PICSS	DBpedia	Cross-Domain	MovieLens Netflix datasets RMSE, Mean Absolute Error MAE, P, R, F measure	1- Freebase 2- LinkedMDB 3- YAGO will be used in future. 4- Utilize users' social relationships
[24]	**Hsu and Lin 2020**	Knowledge-based RS	Automatically build an ontology & recommends related OGDs	1- OWL-based ontology 2- Jena-based rule 3- K-means 4-KNN	- Extract from Facebook - Taiwan government datasets	Government dataset for health care, food safety, and environment protection	Computation time.	1- Develope various domain classes to expand the ontology. 2- Create a chatbot to interact with a user

1.3 Problem Statement

The most recurrent issue in previous research is the lack of semantic information, which pertains to the need to exploit the rich semantics of information about items in addition to the cold-start and data sparsity problems which are associated with collaborative recommendations. The use of knowledge bases is very important to overcome the problems of traditional RS. The LOD cloud is an important knowledge source to obtain various descriptive features to represent the items. Information about the items are freely available and accessible in the LOD cloud as a RDF format and can be obtained by executing queries on SPARQL endpoints.

Most previous research depends on pure link-based similarity measures which will degrade the accuracy of recommendation if there are insufficient links among the resources, even though they share many features. Furthermore, the values of the features in the LOD are stored as literals or non-literals and some approaches consider the lexical content, such as the literals of the items to be the features of the items. However, literals may not be properly formatted or have invalid values that affect the quality of the selected features, for example, the value of the object could be a specific date like “01-01-1550^^xsd:date” or a string value like “16th century”.

On the other hand, some features such as actor, genre, director or writer may be missing in one dataset but available in other. Additionally, some datasets are rich in hierarchical levels like DBpedia, so we can obtain the related resources through Dublin Core Metadata Initiative terms (DCMI terms) and Simple Knowledge Organization System (SKOS) using the properties (dcterms:subject/ skos:broader). However, some datasets do not include such relations. Therefore, if one dataset has insufficient features, it is beneficial to extract the features of specific resources from more than one dataset. However, sometimes the links

between the same resources in two datasets are missing and it is necessary to generate these links to have all the features from the all chosen linked datasets. Sometimes there are links to deprecated dataset and such links have important relations needed that reflect specific knowledge, therefore, a preprocessing is required to find alternative links.

1.4 Objectives

1. Designing a linked data-based RS for multi-domain and proposing an algorithm that depends on the Linked data and utilizes the transversal relations that exist in LinkedMDB and DBpedia datasets as well as the categorical structure that exists in DBpedia in order to find the candidate resources for the recommendations from movie domain and other related domains (Books and Music).
2. Proposing an Extended LinkedMDB dataset that includes extra "**sameAs"** links for fixing the missing links between the similar movies in LinkedMDB and DBpedia using the Silk framework and also extra links from the movies to the related books in British National Bibliography (BNB) Library dataset will be included.
3. Employing word embedding to represent the item's non-literal features and generate a vector for each resource, as well as FOAF (Friend of a Friend) vocabulary to represent the user's preferences by analyzing the user's history and fetching the features required from LinkedMDB and DBpedia after fixing the missing links between those datasets using the Silk framework.
4. Proposing a modified LDSD measure called Enhanced LDSD (ELDSD) that increases the effect of having direct links.
5. Proposing an algorithm to rank the candidates according to scores that are calculated by implementing an Enhanced LDSD If the resources are well

linked or using the cosine distance of the vectorized features if the resources are poorly linked.

1.5 Contributions

The purpose of this dissertation is to contribute to this field of study by the following:

1- Extended version of LinkedMDB dataset is proposed that includes extra SameAs links to DBpedia and new links to BNB Library to link the movie to the related book in BNB Library instead of the depreciated Book Mashup dataset. This will help other researchers in different applications who need to utilize the features of the movies and to find the corresponding DBpedia URI from the LinkedMDB URI.

2- A well-known similarity measure called LDSD is enhanced and a variant of this measure, we call it ELDSD is proposed. ELDSD is recommended to be used in future by other researchers to find the similarity of the resources instead of LDSD.

3- A hybrid semantic similarity approach is proposed and it is recommended to be used by the researchers to find the similarity of resources especially, in some cases when there are very few links between the resources.

1.6 Dissertation Organization

There are five chapters in this dissertation, every chapter begins with a background overview about the related subject and highlight the key characteristics of the chapter. The remaining chapters discuss extra details as the following:

Chapter2 shows a general overview of linked data, RSs and their problems. In addition to a discussion about some techniques used for representing users and items, querying LOD, and calculating the similarity of LOD resources.

Chapter3 discusses the proposed approach in details and illustrates the architecture of all parts implemented to achieve the goal of this dissertation.

Chapter4 presents the experiments done to evaluate the proposed approach and discusses the results obtained.

Chapter5 concludes the dissertation, highlights the findings, and outlines the limitations and the possible future works to develop such systems.

CHAPTER TWO

Theoretical Fundamentals

2.1 Introduction

Over the years, Recommendation Systems (RSs) have been tested and developed, and they are now used in a variety of areas, including social media, video platforms, e-commerce platforms, news, book, tourism, and online advertising [6, 25]. The most crucial objective of a RS is to assess the user's interest about the system's items. This kind of systems produces a list of items for each user according to his interests, as well as an estimate of how much they would like each item. As a result, these systems assist users in locating items of interest and avoiding wasting time in obtaining pertinent information [26, 27]. LOD is a relatively new subject area with enormous promise in a variety of disciplines. For several applications, the usage of LOD has offered huge advantages including discoverability, reusability, accessibility, transparency, and interoperability [28]. Many data-intensive applications based on LOD have been developed in recent years, and RS is one of the technologies that has profited the most. Using LOD for recommendation seems to be quite effective[29].

In this chapter, the main aspects related to RSs and their common issues, the semantic web and Linked Data, the use of LOD for recommendation task, and the main techniques used to find the related resources in LOD will be discussed intensively.

2.2 The Recommendation System Problem

Most algorithms are developed for just one of these two types of recommendation problems: rating prediction or item ranking (also known as top-N recommendation) [30, 31]. Until the Netflix [32] grand prize is awarded, The recommendation problem was known as a rating prediction challenge in the scientific community. In a nutshell, the goal was to estimate a similarity value for

an untested items based on the user's previous experiences. However, in real-world RS implementations, users will see a tiny sample of relevant items. The goal of a top-N recommendation is to provide a selection of N top-ranked items that will be of interest to the user [33]. According to several studies, the rating prediction task was not able to provide the best top-N suggestion lists [34]. As a result, the recommendation problem was re-defined as a top-N recommendation task and the optimization goal became items ranking. The former goal is described as maximization of prediction accuracy, while the latter is oriented on providing the best possible list to the user. The task of recommending the top-N items is also known as the Item recommendation task [35] because, the optimization problem is shifted to items instead of rating. However, the significant distinction is that the value associated with the user-item combination in rating prediction task is an assessment of the rating. Whereas, in the Top-N task, the value is an assessment of a value associated with the item's rank and independent of the related rating [34].

2.3 Recommendation System Categories

RSs are classified into five categories—Content-based, Collaborative, Knowledge-based, Linked data-based and Hybrid approach [21, 36, 37]. Detailed explanation and the main advantages and disadvantage of each category are provided in the next subsections.

2.3.1 Collaborative-filtering

Collaborative filtering techniques rely exclusively on gathering and analyzing massive quantities of data on users' activities, interests, and preferences, and then predicting what people will appreciate based simply on their taste similarity [38]. This RS merely necessitates the development of a factorization model by requiring the rating matrix R. R is a two-dimensional matrix includes **n** users and **m** items;

each value in this matrix, reflects the rating given to a specific item by a specific user. However, it is affected by the issue of cold-start, which happens when a user is freshly registered and the system has no prior knowledge of their preferences, and data sparsity, which occurs when the interaction between the user and items is limited and the ratings are insufficient, resulting in a reduction in the accuracy of recommendation [13, 37, 39, 40].

2.3.2 Content-based RS

This sort of RS is mostly dependent on the content of previously assessed items by specific users. It recommends items that are relevant to the user based on the item's characteristics [41-43]. In contrast to collaborative filtering, content-based filtering does not suffer from cold-start problems, which occur when new items are recommended prior to a sufficient number of users assigning ratings. This technique will match a keyword-based item's attribute to the user profile by using a heuristic method that ignores the semantics of the item characteristics [44, 45]. However, content-based RSs have the well-known drawback of limited content analysis [4], where there are limited or no features describing the items to be recommended. Additionally, it offers a low degree of novelty since it fits the characteristics of the items with a specific user's profile [37].

2.3.3 Knowledge-based RS

This kind of RS analyzes the user's requirements and the item's characteristics obtained from a knowledge base and infers the relationships between them [46]. Aggregating knowledge graphs, such as ontologies, into the recommendation process is one way for overcoming the problems of traditional RSs [47, 48]. Ontology-based (OB) recommenders are knowledge-based RSs that make use of ontologies to express knowledge about items and users throughout the

recommendation process [47]. A knowledge-based RS does not require the collection of a significant quantity of data and helps to solve cold-start problem, since its recommendations are independent of the user's ratings. One of the main disadvantages is the possible knowledge acquisition bottleneck that may occur as a result of explicitly specifying recommendation knowledge [37, 49]. Another disadvantage that precludes the broad use of such systems is a scarcity of data on a large proportion of items, which may vary between domains and change over time [50]. However, this problem is resolved with the existence of cross-domain datasets in LOD cloud and encouraged researchers to work on this kind of RSs.

2.3.4 Linked Data-based RS

In content-based approaches, the information exploited by recommendation engines is very often encoded by bag of words or, more recently, by word embeddings. In all these cases, no explicit semantics is associated to the contextual data. Nowadays Linked Open Data (LOD) datasets represent a huge repository of different kinds of knowledge [51].

Linked Data (LD) is a collection of best practices for sharing and interlinking structured data and knowledge on the Internet by utilizing standard web technologies [52]. RS may leverage information gathered from various datasets, to create a user model and enrich the resources with extra features to enhance the recommendation [53]. In the LOD cloud, an amount of information about the items are publicly available in RDF format and may be retrieved by querying the SPARQL endpoint [17]. Since linked data reflects information from many domains, offers uniform access to data, and shows semantic connections between different entities, it will definitely enhance the RS and overcome the common problems of RSs. This kind of RS is referred to as linked data-based RS. In contrast to traditional knowledge-based RS, linked data-based RS makes use of

datasets created, modeled, and maintained by a variety of organizations and communities worldwide [46].

2.3.5 Hybrid RS

This RS combines several approaches to overcome the limitations of the different techniques when used individually. Collaborative filtering, for example, suffer from the user cold-start problem, which occurs when new users have little or no ratings. On the other hand, content-based prediction avoids this difficulty since it depends on the items' characteristics, which is generally accessible but sometimes struggles with the item cold-start problem. As a result, these concerns may be addressed by merging the two techniques into a hybrid RS [16, 46, 54].

2.4 Problems of Recommendation Systems

The main drawbacks of filtering-based recommendations include poor computational efficiency caused by the sparsity of the data, overspecialization leading to a lack of novelty and serendipity, and the cold-start problem. In particular, CBF strategies are also expected to analyze a wide body of textual, unstructured material in order to obtain knowledge about the items. Due to the ambiguity of natural language processing, they suffer from limited content analysis [55, 56].

The main issues regarding the production of accurate recommendations can be summarized in the following subsections:

2.4.1 Limited content analysis

Content-based RSs are constrained by the characteristics of the suggested items. Thus, in order to acquire a sufficient number of characteristics, the features must

be in a format that can be automatically processed or the characteristics must be manually assigned. Content-based RSs are also confronted with another issue, which occurs when two distinct items with identical properties are indistinguishable by the system [57].

2.4.2 Cold start, data sparsity and "gray sheep"

Cold start and data sparsity are two frequent issues that are associated with collaborative filtering. When inadequate data or metadata is provided, RS performs sub-optimally. On the other hand, cold-start may be divided into two separate subcategories: item cold start and user cold start. When a new item is presented to the system, it undergoes an item cold start, which results in the absence of reviews owing to a lack of user engagement. When a user launches a new account and does not have any previous history, another problem called user cold-start will be occurred. The data sparsity is due to the users' intention to rate a small number of items and the user–item matrix usually contains unknown ratings due to a lack of knowledge to evaluate items and rate them [37]. The most popular solution to these problems is to include auxiliary information to the recommendation algorithm, which may improve the mining capabilities of recommendation algorithms and successfully compensate for sparse or missing interaction information. [58]. On the other hand, in Collaborative filtering and content-based, there is a special case when users do not fit in any taste cluster; this problem is termed as "gray sheep" [59, 60].

2.4.3 User and domain dependence

In certain scenarios, RS may need users to conduct manual actions in order to gather data about their preferences. Such activities include user input, rating, searching, adding useful metadata, and annotating the items. On the other side,

domain dependency arises when recommendations are helpful just for items within a narrowly defined domain without taking into account data from adjacent domains [21].

2.4.4 Data Quality

The data accuracy is affected whenever the source data for items to make recommendations does not include accurate information. Data quality issues may vary, ranging from dependability problems (such as flawed linkages between concepts or erroneous representations) to poor recommendations [21]. Heinrich, et al. [61] investigated the impact of data quality on rating prediction and found that, when more attributes and feature values are supplied for the items, the rating predictions become substantially more accurate.

2.4.5 Privacy

To provide personalized recommendation, a recommendation algorithm needs to get an information from the users, which is a potential security and privacy concern. However, it is important and desirable to offer a RS that do not need to access personal data, while still building robust and accurate recommendation models [57, 62].

2.4.6 Scalability and computational complexity

To provide recommendations that are relevant to a specific user, the RS is required to use a massive amount of data to run their algorithm. Scalability issues have increased because of the speed at which e-commerce websites have expanded. The new RS methods that are needed to work with large scale applications must provide results quickly. RSs can locate several possible neighbors at a moment,

but for contemporary e-commerce sites, they need to locate more. Dimensionality reduction, Bayesian Network, and Clustering are all tools that may be used to help address scalability issues [21, 37, 57]. In addition, RSs can be applied in cloud environment to avoid scalability problem [63].

2.4.7 Diversity, Novelty, and Serendipity

It is possible for a RS to give recommendations for similar or more diverse items. When suggesting items based on user or item similarity, the results are most accurate, but the problem of diversity arises. As a result, the user is exposed to a small number of items, and the most relevant items may be missed. On the other hand, if an algorithm is only increases diversity, the accuracy will decrease. As a consequence, it's critical to make sure that the recommender results cover as much of the item space as possible and are not all originated from the same cluster [37, 64]. On the other side, novelty refers to proposing goods that have novel elements and are unfamiliar to the user, while serendipity is not only unfamiliar, but also surprises the user. [64, 65].

2.4.8 Overspecialization Problem

Recommendations will be provided to the users based on those items that are previously known to them or specified in their user profile, without finding new items or other available options [64].

2.4.9 Synonymy

It is a scenario in which two or more items are similar but have distinct names or entries. Closely similar items such as "comedy movie" and "comedy film" cannot be distinguished using RS algorithms. The extensive use of synonymous terms degrades the collaborative filtering recommendation's

performance. There are some techniques that can be used to overcome this issue such as (a) Singular Value Decomposition (SVD), (b) Latent Semantic Indexing, and (c) Construction of thesaurus [57]. Certainly, this issue could also be solved by using LOD and utilizing the different kinds of existing ontological relations that have different hierarchical levels.

2.5 Semantic Web

The World Wide Web (WWW) makes it feasible to produce online content and to make current material accessible in digital forms. Berners-Lee, et al. [66] claim that the semantic web was created to unify online content by tagging it with unique identifiers, or Uniform Resource Identifiers (URIs), representing it with well-formed definitions from taxonomies and ontologies, and borrowing from the world of knowledge representation to use data structuring mechanisms [67]. The semantic web is a modification of the present web that allows users to efficiently exchange, search, integrate, and reuse data. Basic web data must be translated into structured data using Resource Description Framework (RDF). Therefore, it presents data in machine readable language and this is the fundamental goal of the semantic web to allow computers to search for information without the need for human intervention. RDF is a suggested standard for describing metadata defined by online sites. To tackle the interoperability issue, which is the ability of computers to use, exchange, and understand the information, RDF and other semantic web technologies like RDF Schema (RDFS) and Web Ontology Language (OWL) can be used [68] and they will be discussed in the next subsection. However, the vision of a semantic web expressed in several ways that all share one aim of creating a web of machine-readable data [2]. According to Bizer, et al. [69] “The term Linked Data refers to a set of best practices for publishing and connecting structured data on the Web. While the semantic web, or

web of data, is the goal or the end result of this process, Linked Data provides the means to reach that goal".

2.5.1 Basic Technology for the Semantic Web

The architecture of the semantic web is guided by three principles. The first is the establishment of standard formats for structured and semi-structured data. The second is to make datasets and each piece within them, along with their relationships, accessible over the web. The third is to incorporate semantic into the data. The semantic web's fundamental technology is summarized by Groth, *et al*. [1] as follows:

1. Utilizing labeled graphs as the data model for resources and their relations, with nodes representing the resources and edges representing the relationships between them and RDF is used to express such graphs.
2. Making use of Uniform Resource Identifiers (URI) to identify every resource and every relation in the datasets to distinguish every data item with a unique ID. Again, the design of RDF reflects this.
3. To explicitly capture the intended semantics of the data, use ontologies as the data model. For this aim, formalisms such as RDFS and the OWL are employed.

Each fact (statement) in RDF is represented as a collection of triples (subject, property/predicate, object), indicated by (s, p, o), where subject (s) is an entity, class, or blank node, property (p) is one attribute associated with one entity, and object (o) is an entity, a class, a blank node, or a literal value. An entity is identified by a URI that relates to a named resource in the modelled environment while the blank nodes, refer to anonymous resources without a name. As a result, each triple represents a named connection; blank nodes means that a relationship exists and associated with something that is undefined. RDFS and OWL, both of

them are W3C standards used to annotate RDF data with semantic information. This annotation is mostly used to reason about RDF data and infer new knowledge about the data, this is also called entailment [70, 71]. RDFS and OWL are lightweight knowledge representation languages, not only data-description languages. RDFS provides a way to describe the classes of RDF resources, their hierarchy, the relations among the different resources and the hierarchy of these relations. Given an RDF graph describing a set of resources and an RDFS ontology describing the classes and relations, RDFS entailment rules generate new facts that extend our knowledge about the resources. Similarly, OWL is a more complex (but still lightweight) logic that allows for additional inferences like equality and inequality, number restriction, object existence, and so on [1, 72].

2.6 Linked Open Data (LOD)

Linked Data is a vast network of linked nodes that is often referred to as the Giant Global Graph (GGG) [11, 73]. LOD is a community initiative aimed at providing free and public data. The datasets in LOD are accessible on the web under an open license in machine-readable format (RDF triples) and are linked together with specific URIs [50]. Linked Data is based on two web-oriented technologies: Uniform Resource Identifiers (URIs) and the HyperText Transfer Protocol (HTTP). While Uniform Resource Locators (URLs) have become popular as addresses for documents and other resources that can be found on the Web, URIs provide identification to any entity that exists in the world [69]. Thus, certain modifications to the hypertext Web are required to enable the Web of data to define resources and publish data. As illustrated in Figure 2.1, this entails two steps: expanding the set of exchange formats beyond HTML (and other document formats) to include the Resource Description Framework (RDF) and its serializations for data exchange, and generalizing URLs (Uniform Resource Locators) to URIs (Uniform Resource Identifiers) or IRIs (Internationalized

Resource Identifiers), which can be used to identify and then describe anything [74].

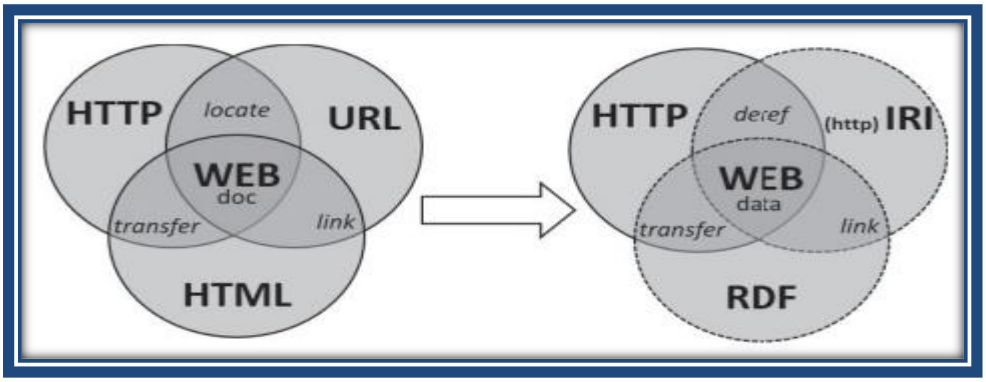

Figure 2.1 Relying on the Web architecture to publish linked data[74].

The purpose of Linked Data is to establish a huge Graph of knowledge. However, to describe the linked data theoretically, Meymandpour and Davis [11] give the following mathematical definitions for the Linked Data, its features, and its resources.

Definition 1. Linked Data (LD) is a labeled directed graph, defined as (R, L, T), such that R = $\{r_1, r_2, \ldots, r_{|R|}\}$ is a set of resources (nodes, vertices), L = $\{l_1, l_2, \ldots, l_{|L|}\}$ is a set of links (edges, relations, predicates) and T = $\{t_1, t_2, \ldots, t_{|T|}\}$ is a set of triples (statements) such as (r_1, l_1, r_2), where $l_1 \in L$ is a link from $r_1 \in R$ to $r_2 \in R$.

Definition 2. According to Meymandpour and Davis [11] "A feature f of the resource $r \in R$ in Linked Data (LD) is denoted as a triple of kind (l, r_t, D), where $r_t \in R$ is the (target) resource directly connected to r via the link $l \in L$ and D is the direction of the link (In/Out)".

Definition 3 Also, according to Meymandpour and Davis [11] "(A Resource in Linked Data). A resource $r \in R$ in Linked Data (LD) is denoted as a set of its features F_r , defined as follows:"

$F_r = F_r^{Out} \cup F_r^{In}$

$F_r^{Out} = \{\forall(l_i, r_i, Out), l_i \in L, r_i \in R \mid \exists(r, l_i, r_i) \in LD\}$

$F_r^{In} = \{\forall(l_i, r_i, In), l_i \in L, r_i \in R \mid \exists(r_i, l_i, r) \in LD\}$

The graph in Figure 2.2 represents an example of RDF triples. Every node represent a specific Resource (R) that is linked to another node (Non-Literal) or linked to Literal.

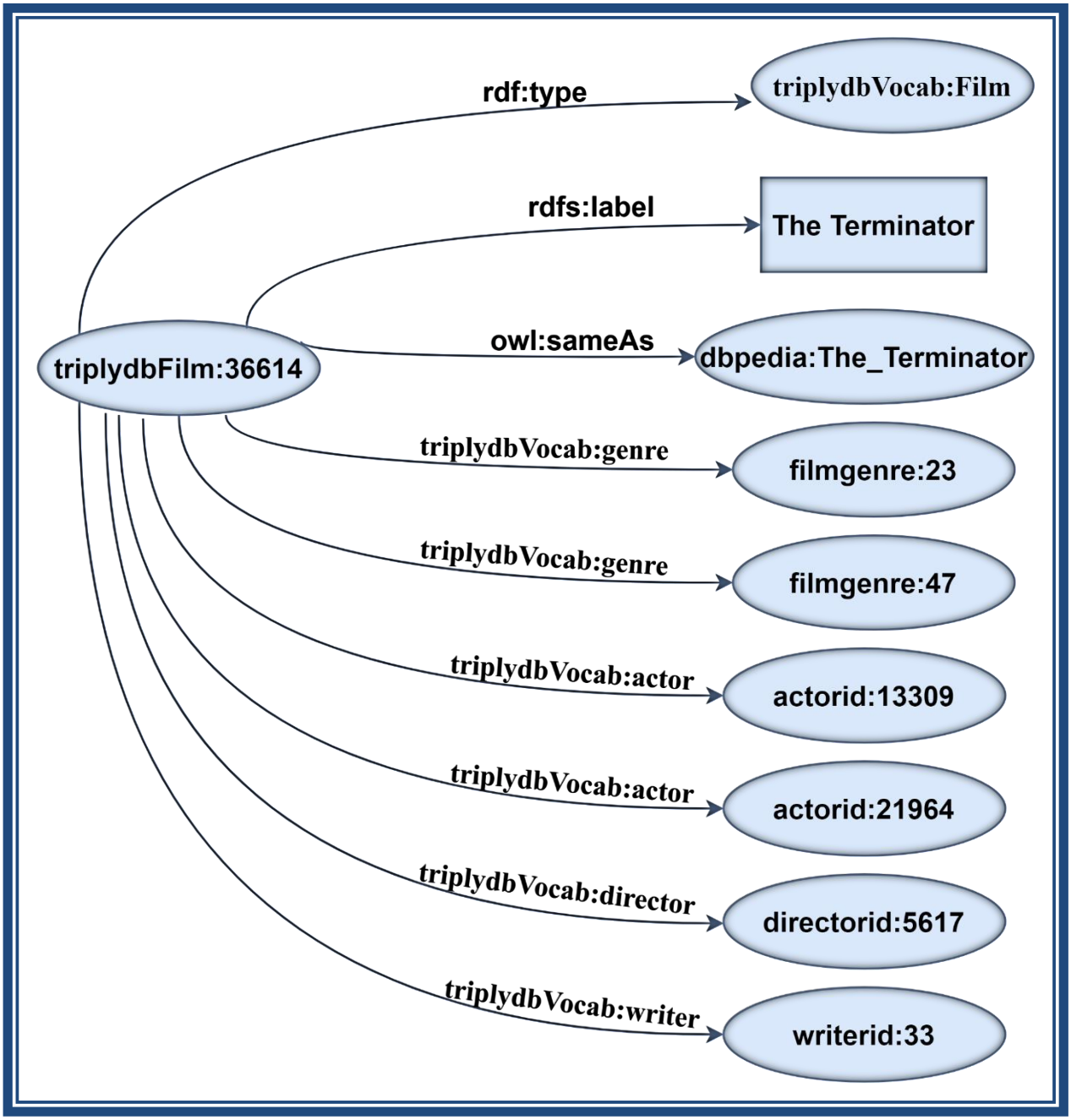

Figure 2.2 An example of graph representation of RDF triples

The prefixes mentioned in Figure 2.2 represent specific namespaces as defined in Table 2.1 :

Table 2.1. Prefixes and their namespaces

Prefix	namespace
triplydbFilm	**“**https://triplydb.com/Triply/linkedmdb/id/film/”
triplydbVocab	**“**https://triplydb.com/Triply/linkedmdb/vocab/”
writerid	“https://triplydb.com/Triply/linkedmdb/id/writer/”
directorid	“https://triplydb.com/Triply/linkedmdb/id/director”
actorid	“https://triplydb.com/Triply/linkedmdb/id/actor”
filmgenre	“https://triplydb.com/Triply/linkedmdb/id/film_genre”
dbpedia	**“**http://dbpedia.org/resource/”
rdf	“http://www.w3.org/1999/02/22-rdf-syntax-ns#”
rdfs	“http://www.w3.org/2000/01/rdf-schema#”
owl	“http://www.w3.org/2002/07/owl#”

RDF triples can be defined as (R, L, T) as shown in Figure 2.3:

R= { <https://triplydb.com/Triply/linkedmdb/id/film/36614>,
<http://dbpedia.org/resource/The_Terminator> ,
<https://triplydb.com/Triply/linkedmdb/id/actor/13309> ,
<https://triplydb.com/Triply/linkedmdb/id/actor/21964> ,
<https://triplydb.com/Triply/linkedmdb/id/actor/23377> ,
<https://triplydb.com/Triply/linkedmdb/id/director/5617>,
<https://triplydb.com/Triply/linkedmdb/id/film_genre/23> ,
<https://triplydb.com/Triply/linkedmdb/id/film_genre/47> ,
<https://triplydb.com/Triply/linkedmdb/id/film_genre/99> ,
<https://triplydb.com/Triply/linkedmdb/id/writer/8820> ,
<https://triplydb.com/Triply/linkedmdb/id/writer/9111> ,
<https://triplydb.com/Triply/linkedmdb/id/writer/9447> ,
}

L={
<http://www.w3.org/2000/01/rdf-schema#label> ,
<http://www.w3.org/2002/07/owl#sameAs> ,
<https://triplydb.com/Triply/linkedmdb/vocab/actor> ,
<https://triplydb.com/Triply/linkedmdb/vocab/director> ,
<https://triplydb.com/Triply/linkedmdb/vocab/sequel> ,
<https://triplydb.com/Triply/linkedmdb/vocab/genre> ,
<https://triplydb.com/Triply/linkedmdb/vocab/writer>
}

T ={
< <https://triplydb.com/Triply/linkedmdb/id/film/36614> ,
<https://triplydb.com/Triply/linkedmdb/vocab/actor>,
<https://triplydb.com/Triply/linkedmdb/id/actor/13309> **>,**

< <https://triplydb.com/Triply/linkedmdb/id/film/36614> ,
<https://triplydb.com/Triply/linkedmdb/vocab/actor>,
<https://triplydb.com/Triply/linkedmdb/id/actor/21964> **>,**

< <https://triplydb.com/Triply/linkedmdb/id/film/36614> ,
<https://triplydb.com/Triply/linkedmdb/vocab/actor>,
<https://triplydb.com/Triply/linkedmdb/id/actor/23377> **>,**

< <https://triplydb.com/Triply/linkedmdb/id/film/36614> ,
<https://triplydb.com/Triply/linkedmdb/vocab/director>,
<https://triplydb.com/Triply/linkedmdb/id/director/5617> **>,**

< <https://triplydb.com/Triply/linkedmdb/id/film/36614> ,
<https://triplydb.com/Triply/linkedmdb/vocab/director>,
<https://triplydb.com/Triply/linkedmdb/id/director/5617> **>,**
.....
}

Figure 2.3. RDF triples Definitions

2.6.1 LOD Datasets

LOD cloud [75] contains multiple datasets, and one of the most extensive datasets is DBpedia, which is a dataset containing all types of knowledge including a number of entities and their relationships [76, 77]. It is community efforts to retrieve structured material from Wikipedia and make it publicly available as RDF triples [76]. For the purpose of discovering more knowledge from the DBpedia dataset, it is interlinked with numerous other data sources on the web via RDF links, for example, it is interlinked with LinkedMDB using 13800 outgoing links, while the ingoing from LinkedMDB is 30354 links. However, the total number of DBpedia links to all linked datasets is 39,007,478. DBpedia URIs can also be used to convey personal preferences, places of residence, and related information inside personal FOAF profiles, e.g. ***foaf:topic*** is used to relate to a particular subject that exists in DBpedia. A more commonly used case is the categorization of blog articles, news reports, and other documents. The benefit of this method is that all DBpedia URIs are supported with details and therefore enable clients to retrieve more knowledge about the searched subject [78].

In 2007, there was only 12 datasets as shown in Figure 2.4, while the 2022 edition of the LOD cloud, shown in figure 2.5 includes 1301 datasets with 16283 links. Each node (circle) corresponds to a dataset. The different colors show different domains of application (for example, life sciences, publication, government, geography, ... etc.).

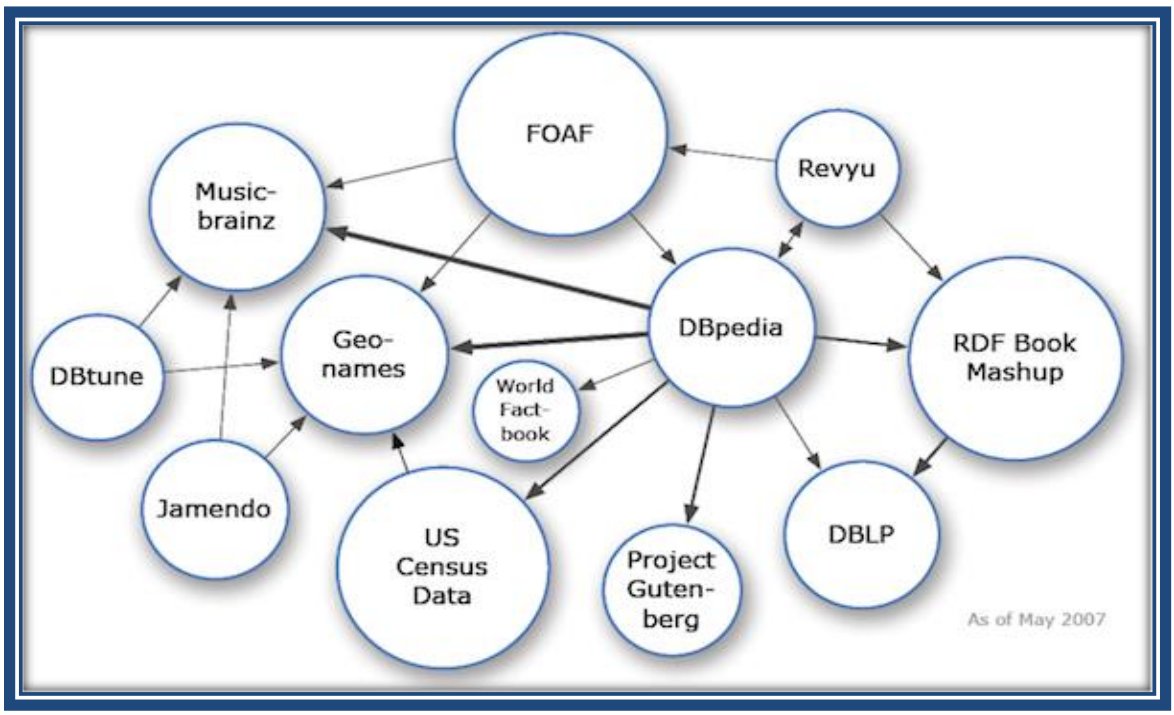

Figure 2.4. LOD Cloud in 2007[79]

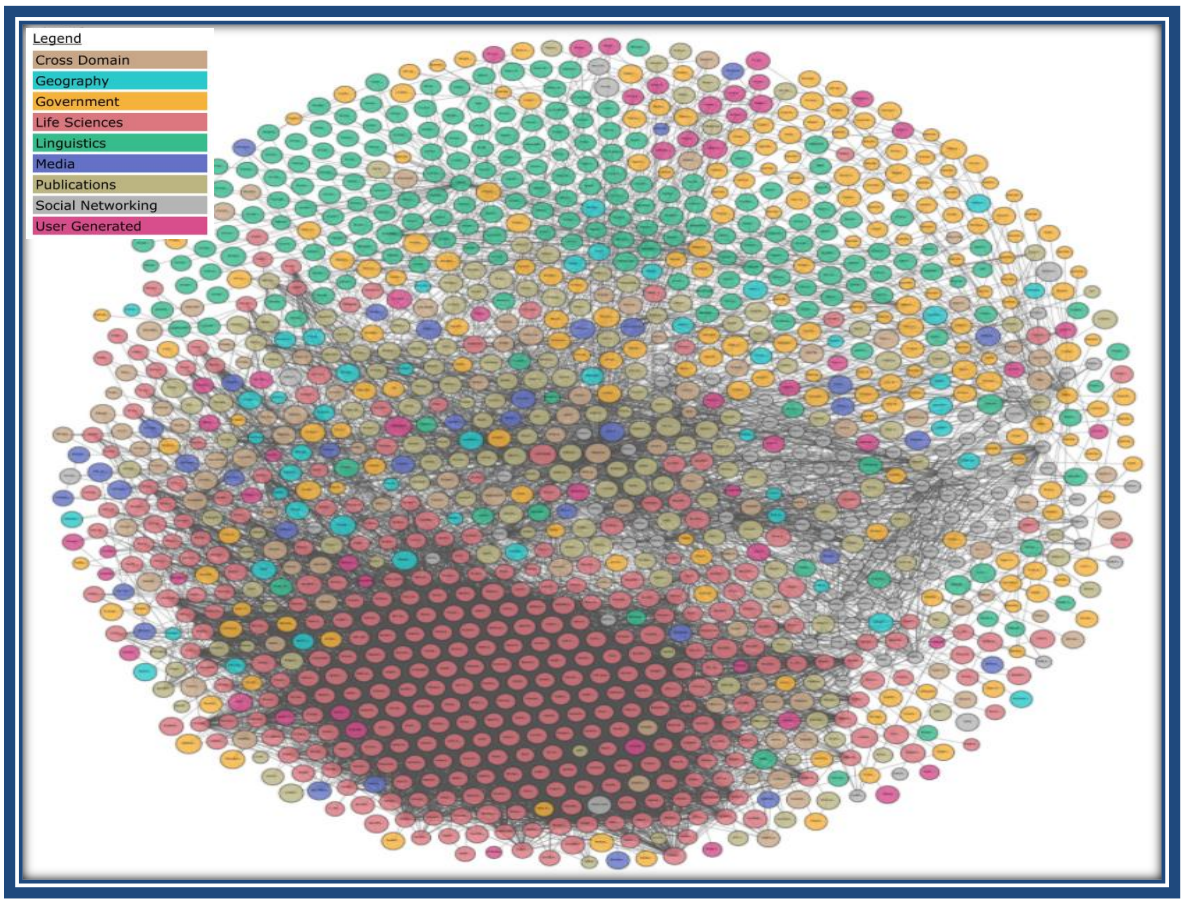

Figure 2.5. LOD Cloud in 2022[75]

2.6.2 SPARQL for Querying Linked Data

Simple Protocol and RDF Query Language (SPARQL) is used for querying a Linked data. It's quite similar to SQL and may be used in the same manner. However, there are significant distinctions. To begin with, the data is not saved in tables, but rather in triples that make up a graph. Second, by using Internationalized Resource Identifiers (IRIs) as universally unique identifiers, you can forget about joins and foreign keys. Another significant distinction is that the data structure and data itself are both expressed as triples. This implies that the same query language may be used to query both the data structure and the data [80].

The most common software to implement a SPARQL query is called a *triple store*. Essentially, triple stores are databases for RDF and most of them provide bulk upload options to upload RDF data. After data has been put into a triple store, it may be queried by utilizing the SPARQL protocol. Each triple store has an endpoint, which is a place where SPARQL queries may be sent. Clients use the HTTP protocol to submit queries to an endpoint. Because SPARQL is based on conventional web technologies, there are many SPARQL endpoints available on the internet. These endpoints provide you access to a lot of data. For instance, DBpedia [75] offers a query endpoint for querying over Wikipedia's RDF representation [1].

As an example, if a RDF describing *The Terminator* movie as shown in Figure 2.6 A, a query can be executed on this information and find the writers of The Terminator movie. Any member of the triple can be replaced with a variable in SPARQL. Variables are denoted by a (**?**) at their beginning as shown in Figure 2.6 B.

A

```
@prefix dbpedia: <http://dbpedia.org/resource/>.
@prefix dbont: <http://dbpedia.org/ontology/>.
dbpedia:The_Terminator dbont:writer
                       dbpedia:James_Cameron,
                       dbpedia:Gale_Anne_Hurd
```

B

```
PREFIX dbpedia: <http://dbpedia.org/resource/>.
PREFIX dbont: <http://dbpedia.org/ontology/>.
SELECT ?writers
WHERE {
      dbpedia:The_Terminator dbont:writer ?writers
}
```

Figure 2.6. Example of RDF triples and SPARQL query

The query finds triples where *dbpedia:The_Terminator* is in the subject position and *dbont:writer* is in the predicate position.

To begin, all prefixes must be specified. The **PREFIX** keyword stands for numerous URL abbreviations. The term **SELECT** specifies which variables are of interest. After that, the **WHERE** clause refers to the graph pattern that has to be matched. The query's results are delivered as a collection of bindings, which indicate which items relate to which variables. Each row is a single binding or outcome [1]. Thus, for a result of the above query we would get the following bindings for the variable *writers*:

?writers
"http://dbpedia.org/resource/Gale_Anne_Hurd"
"http://dbpedia.org/resource/James_Cameron"

2.6.3 Linked Data for Recommendation System

Several works that address recommendations in numerous domains have been suggested in literature, even though very few methods take advantage of the LOD initiative to provide efficient recommendations. LOD was often used to mitigate the cold-start and data sparsity issues associated with CF[81]. Linked data-based RS is a knowledge-based RS, whereby the knowledge base is represented as tuples of resources, categories and links. The Resources are instances of concepts (thoughts or notion) which are identified by the URI. The Categories refer to the types or classes. Finally, the links are known as relations or properties linking nodes to others. There are three kinds of relationships; ***Resource-Resource (R-R), Resource-Category (R-T)*** and ***Category-Category (T-T)***.

R-R relationships are the transversal relationships between resources and do not refer to hierarchical classifications. Much of the DBpedia resources have these kinds of relations with each other. The second is ***R-T*** relationships which are the relationships connecting a resource to the category. They may be expressed using the ***rdf:type*** property. DBpedia interlinks resources to its Wikipedia categories using ***dcterms:subject***, the Dublin Core vocabulary term. The third is ***T-T*** relationships which are hierarchical relationships between categories within a hyponymous system (a category tree). They can be interpreted using the RDFS property ***rdfs:subClassOf*** or the SKOS properties ***skos:narrower*** (**is-SuperCategoryOf**) and ***skos:broader*** (**is- SubCategoryOf**)[46].

The previous studies are categorized according to how Linked Data are used to make recommendations into four categories. **First**, the **Linked Data-driven RS** depends on the knowledge expressed as Linked Data to produce recommendations. In some RSs, the calculation of semantic similarity depends on the variety of links that can be identified in LOD datasets. These links are considered as features of the items. **Second**, **Hybrid RS** that uses Linked Data to execute such operations

that may or may not be used to make recommendations. **Third, Representation only RS** uses the RDF representation to describe data and uses one or more ontology to convey the embedded semantics. For instance, some RS utilized FOAF vocabulary to represent the user details, and the Linked Data is not included in any other tasks. **The fourth**, **Exploratory search system**, helps users to discover knowledge and to identify subjects or concepts that are related to an initial concept. This type of RS needs an explicit query provided by the users [21].

2.6.4 User's Preferences Representation

The user profile may be represented using a variety of ontologies. FOAF is the most widely used ontology for modeling individuals [82]. It's used to create models of user's identities, connections, interests, and activities. The core of FOAF is used to describe the user profile. For example, a user entity describes personal information properties such as *foaf:name*, social properties such as *foaf:interest*, foaf:knows and foaf:Group, and personal identities such as *foaf:homepage* and *foaf:mbox* [83]. The RS may model the user's profile using FOAF classes and properties to provide a personalized RS. The user data will then be provided in a semantic manner that may be directly exploited or queried later via SPARQL queries [84] . Figure 2.7. illustrates an example of the user's preferences represented using FOAF vocabulary and used *foaf:interest* property to determine the features preferred by the user.

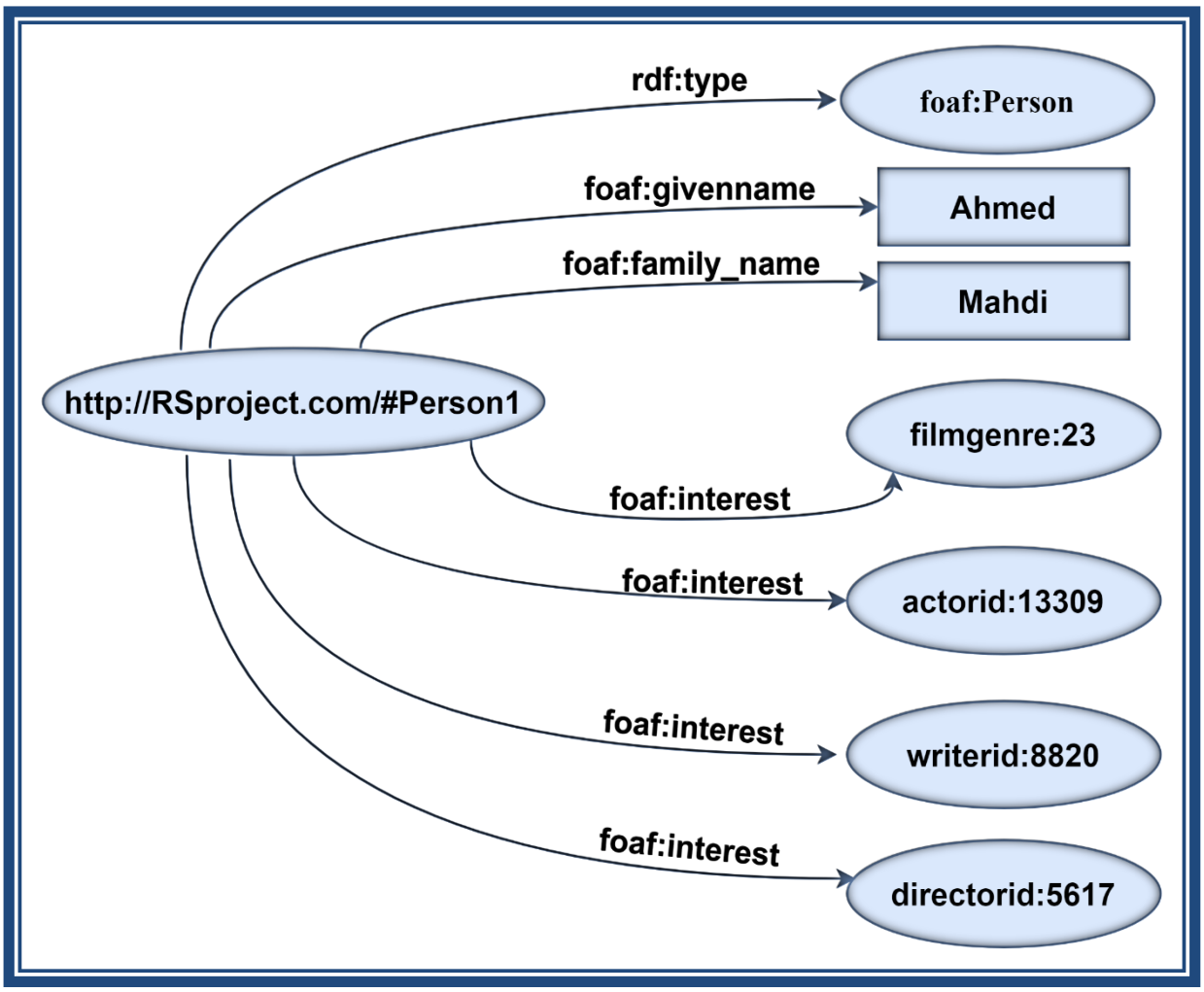

Figure 2.7 Example of FOAF graph to represent user's interests

2.7 Semantic Similarity of Linked Data Resources

Semantic similarity reflects the relationship between the meaning of two resources. It is calculated using a collection of parameters drawn generally from a knowledge representation model. Similarity measurement is essential in a broad variety of Linked Data-based applications, including search result ranking, RSs, and data clustering. There are four types of semantic similarity measures: the first is distance-based models, which are depended on the structural representation of the resources and the existence of relations. The second is feature-based models, which represent resources as sets of features. The third is statistical methods, which

take statistics from the underlying context into account, and the last one is hybrid models, which combine more than one category [11].

In this dissertation and specifically in the next subsection, we will focus on distance-based measures that depend on the link structure of the resources and feature-based measures to find the similarity of the resources in the Linked Data.

2.7.1 Pure Link-based measures

The structure of the graph in LOD is used by researchers to measure the relatedness of the resources by calculating the semantic distance. Those kinds of measures depend on intuition that the more interlinked resources to each other, the more related they are. Linked Data Semantic Distance (LDSD) and Typeless Linked Data Semantic Distance (TLDSD) are examples of the pure link-based measures which are as follows:

A. Linked Data Semantic Distance (LDSD)

This measure is proposed by Passant [8] by depending on the existing of links between the resources. The author proposed a measure that focuses on direct links, another measure focuses on indirect links, and the third measure is a combination of both direct and indirect links. All of them provide a similarity score in [0-1] range, and the resources are considered similar if the distance between them decreases, i.e. the score close to zero. However, the combined measure deals with both types of links, the direct and the indirect, equally without giving importance to either one. The below equation (2.1) shows the combined measure and the score is ranges between 0 and 1.

$$LDSD(r_a, r_b) = \frac{1}{1 + D + I} \tag{2.1}$$

Where D and I are number of direct and indirect links, respectively. Both of them normalized by all outgoing links and can be calculated using equation (2.2) and (2.3).

$$D=\sum_{i}\frac{C_d(l_i,r_a,r_b)}{1+\log(C_d(l_i,r_a,n))}+\sum_{i}\frac{C_d(l_i,r_b,r_a)}{1+\log(C_d(l_i,r_b,n))} \quad (2.2)$$

$$I=\sum_{i}\frac{C_{ii}(l_i,r_a,r_b)}{1+\log(C_{ii}(l_i,r_a,n))}+\sum_{i}\frac{C_{io}(l_i,r_a,r_b)}{1+\log(C_{io}(l_i,r_a,n))} \quad (2.3)$$

where;

$C_d(l_i, r_a, r_b)$: the number of direct links between resources (r_a,r_b) through the direct link(l_i)

$C_d(l_i, r_a, n)$: number of links from the resource r_a to any other nodes through the link (l_i).

C_{io} and C_{ii}: number of indirect links, outcoming and incoming, respectively.

The number of direct and indirect links between the resource r_a and r_b are weighted by depending on the number of links from a specific resource to any other resources in the graph G_i in order to reduce the effect of the most popular links.

B. Typeless Indirect Semantic Distance

In recommender systems, it is necessary to rely on the relationships that exist between the resources, even if they are related to each other through different properties. The LDSD does not take this into account, so Alfarhood, et al. [10] proposed a measure based on the LDSD, but the distance between the resources calculated by considering the number of links that exist even if the links have different properties. The Figure 2.8 and Figure 2.9 illustrates the typeless outgoing

and incoming indirect links, respectively. As can be seen, r_a and r_b are linked directly through a middle resource r_c using different properties l_i and l_j.

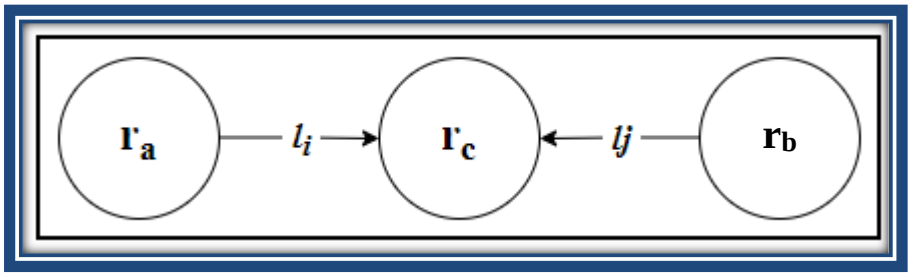

Figure 2.8 Typeless Outgoing Indirect Link

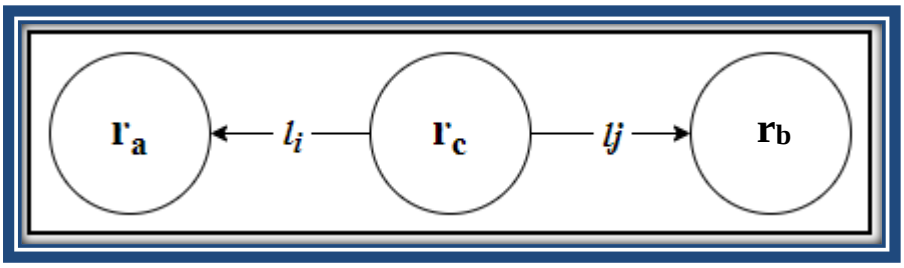

Figure 2.9 Typeless Ingoing Indirect Link

2.7.2 Feature and Content-based measures

This type of measures does not take in consideration the structure of the graph, it just depends on the content or the features in common to find the similarity of the resources. PICSS and EPICS are examples of this type of measures which are as follows:

A. Partitioned Information Content-based Semantic Similarity (PICSS)

Meymandpour and Davis [11] proposed a similarity measure that combines the merits of information content-based measures and the feature-based measures. The measure depends on the information theory concepts that say the frequent features are uninformative while the distinct features lead to more information. The

information content (IC) of a feature in Linked Data can be calculated using the equation (2.4).

$$IC(f) = \log(\frac{1}{\pi(f)}) = -\log(P(f)) \quad (2.4)$$

Where:

$P(f)$: The probability of feature (f) in Linked Data (LD)

The probability of feature (f) is calculated using equation (2.5) that finds the ratio of the number of resources with the feature (fi) to the total number of resources:

$$P(fi) = \frac{Freq(fi)}{N} \quad (2.5)$$

The summation of the information content of each feature (fi) in the set of features (F_r) of specific resource will result the partitioned information content (PIC) of the resource (r) as shown in equation (2.6).

$$PIC(r) = \sum_{\forall fi \in Fr} -\log\left(\frac{Freq(fi)}{N}\right) \quad (2.6)$$

They used this equation in Tversky ratio model[85] to propose PICSS measure that finds the similarity of two resources r1 and r2 represented by two set of features Fr1 and Fr2 respectively using the below equation (2.7).

$$PICSS(r,s) = \frac{PIC(F_{r1} \cap F_{r2})}{PIC(F_{r1} \cap F_{r2}) + PIC(F_{r1} - F_{r2}) + PIC(F_{r2} - F_{r1})} \quad (2.7)$$

Where:

$Fr_1 \cap Fr_2$: Set of Common Features.

Fr_1 - Fr_2: Set of Distinct Features of first resource.

Fr_2 - Fr_1: Set of Distinct Features of second resource.

B. Extended Partitioned Information Content Similarity (EPICS)

Komeiha, *et al*. [12] proposed a variant of PICSS by considering the set of similar features in addition to the same three features used in PICSS that mentioned above. The features are considered similar, if they have same relation type and

same direction. EPICS will be calculated in equation (2.8) using the four set of features defined.

$$EPICS(r,s) = \frac{PIC(F_{r1} \cap F_{r2}) + PIC(F_{r1} \approx F_{r2})}{PIC(F_{r1} \cap F_{r2}) + PIC(F_{r1} \approx F_{r2}) + PIC(F_{r1} - F_{r2}) + PIC(F_{r2} - F_{r1})} \quad (2.8)$$

Where :

$PIC(F_{r1} \approx F_{r2})$ will be calculated as follows:

$$PIC(F_{r1} \approx F_{r2}) = \frac{\sum_{\forall fi \in (F_{r1} \approx F_{r2})} -\log\left(\frac{Freq(fi)}{N}\right)}{2} \quad (2.9)$$

2.8 Semantic Similarity Analysis of Sentences

There are some techniques that have been available for a long time for analyzing the semantic similarity of text such as: traditional semantic vector models, which focused on conveying the word frequency. Using the Term Frequency-Inverse Document Frequency (TF-IDF) formula, it could be found out whether or not a word is relevant to a document. It is computed as the product of the word's frequency in the text and the logarithm of the reciprocal of the word's frequency in the corpus of data. Latent Semantic Analysis (LSA) is another method for determining the relationships between words and a set of texts. It does not consider the context of the word. Singular Value Decomposition (SVD) technique is used to extract a low-dimensional matrix of the word-document co-occurrence, which is subsequently utilized to learn the topic of document. Even though the text has different topics, the Latent Dirichlet Allocation (LDA) approach converts the document to a vector and provides the ability to assign words to each related topic. It also does not consider the context. The absence of common terms in sentences makes those traditional techniques fail in uncovering hidden semantic relationships. On the other hand, the Word2vec model, which is a global word embedding method, that excels in analogy reasoning tasks [86]. This model will train neural networks, allowing it to create a low-dimensional, dense representation

of words with two key characteristics: first, similar words will be close together in vector space, and second, the relationships between the words will be represented as vectors in vector space, allowing us to perform mathematical operations on them [87].

2.8.1 Word2Vec

Word2Vec is a two-layer neural network model for learning word embeddings from corpus of text that is very efficient in terms of computation. Word2Vec is commonly used to handle analogy problems for example, to predict the value of x in this statement: “man is to woman as king is to x”. x is queen, according to Word2Vec. Word2Vec handles analogy problems like this by attempting all of the words in the vocabulary, V, and identifying the one that maximizes the equation (2.10) [88].

$$x = ARGMAX_{x' \in V} sim(x', king + woman - man) \quad (2.10)$$

It is possible to choose one of the two distinct algorithms: the Continuous Bag of Words model (CBOW) and the Skip-gram model as shown in Figure 2.10.

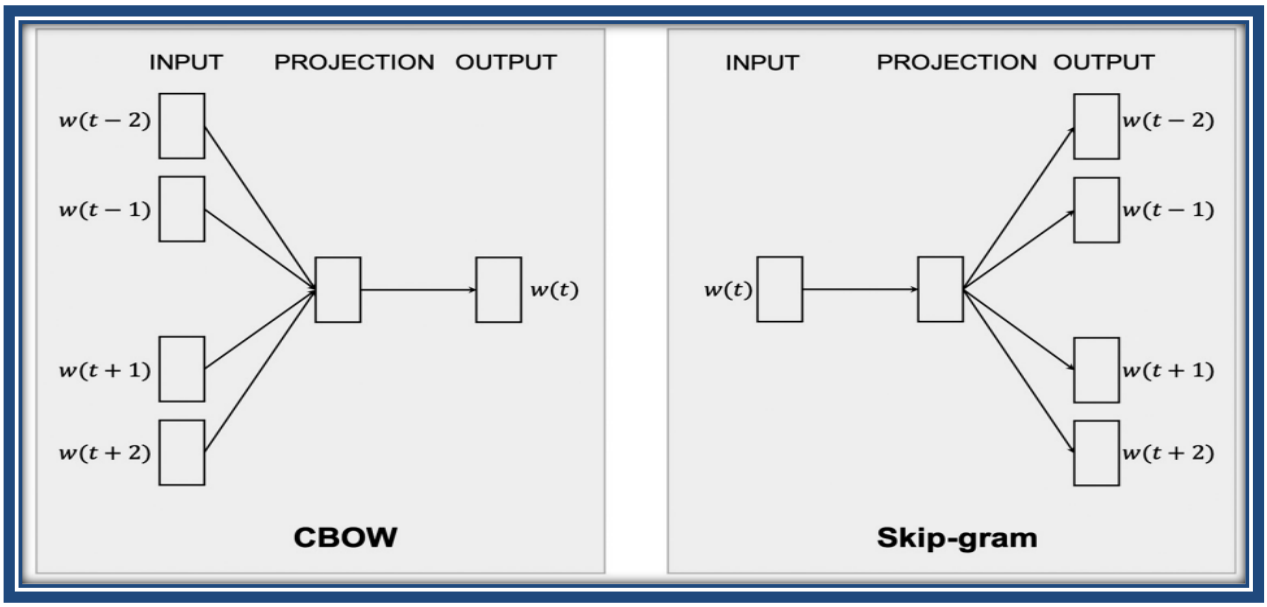

Figure 2.10 CBOW and Skip-gram architecture[89]

A. CBOW model

The CBOW model predicts target words based on context words contained inside a certain window size. The input layer contains all surrounding words for which input vectors are extracted from the input weight matrix, averaged, and projected in the projection layer. After that, a score for each word in the vocabulary is produced using the weights from the output weight matrix. This score represents the probability of the word being a target word. The CBOW model's aim is to maximize the average log probability by the objective function in equation (2.11), when a series of training words $w_1,w_2,w_3,...w_T$ and a context window c are given. [87]

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \log p(w_t | w_{t-c} \dots w_{t+c}) \quad (2.11)$$

Where,

$J(\theta)$ is an objective function, T is the number of training words, and $p(w_t | w_{t-c} \dots w_{t+c})$ is the probability that can be calculated using the following equation which represents a softmax function.

$$p(w_t | w_{t-c} \dots w_{t+c}) = \frac{\exp(v^{-T} v'_{wt})}{\sum_{w=1}^{V} \exp(\bar{v}^{-T} v'_w)} \quad (2.12)$$

Where v'_w is the output vector of the word w, V is the complete vocabulary of words, and $\bar{v}$ is the averaged input vector of all the context words that can be calculated as follows:

$$\bar{v} = \frac{1}{2c} \sum_{-c \le j \le c, j \ne 0} w_{t+j} \quad (2.13)$$

B. Skip-gram model

The skip-gram model does the inverse of the CBOW model and tries to predict the context words from the target words. Given a sequence of training words w1;w2;w3; ... ;wT , and a context window of size c, the objective of the skip-gram model is to maximize the following average log probability[87].

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j}|w_t \theta) \quad (2.14)$$

The probability will also be calculated using the softmax function as follows:

$$p(w_{t+c}|w_t) = \frac{\exp(v'^{T}_{w_{t+c}} v_{w_t})}{\sum_{v=1}^{V} \exp(v'^{T}_{w_v} v_{w_t})} \quad (2.15)$$

Where v_w is the input vector of the word w and v'_w is the output vector. V as previously mentioned, is a complete vocabulary.

The softmax calculation is proportional to vocabulary size, therefore, it is inefficient and optimization approaches are proposed. The first is hierarchical softmax, while the second is negative sampling and the later do better in most experiments done by Mikolov, et al. [90] and used also by Chamberlain *et al*. [91] as it has a better performance than hierarchical softmax. After training is complete, all words are projected onto a lower-dimensional feature space, with semantically similar words clustered together[87, 90].

Skip-gram with negative sampling is a popular choice of Word2Vec that was originally designed for Natural Language Processing. It has been also successfully applied in recommendation tasks. Despite the fact that these fields do not share the same type of data or evaluate on the same tasks, recommendation applications frequently use the same pre-tuned hyper-parameters values, even though the best value for each hyper-parameter is a data and task dependent [92]. Therefore tuning some hyper-parameters such as window size and embedding dimension for recommendation task especially for LOD-based recommendation may improves the performance on this task.

As a summarization of the above, Table 2.1 below shows the hyper-parameters needed in Word2Vec model with their description. The default value for each one and the reference that mention this value is also descibed.

Table 2.2. Word2Vec Hyper-parameters

Parameter	Description	Default value with the reference
Layer Size	Embedding vector dimension	100 [91]
Window size	Sliding window max length	5 [91] [93]
epochs	Number of iteration for training	3-50 [93]
Approximation of Softmax	Methodology to approximate the Softmax	Hierarchical Softmax or Negative Sampling [91]
Negative Samples	Number of negative samples for approximating Softmax	5 [91] 25 [87]
Learning Algorithm	Architectures to learn the underlying word representations	SkipGram or CBOW [91]

Mikolov, et al. [90] find that negative samples range from 5-20 leads to better performance for small training data while range from 2-5 is good for large training datasets. Other researchers [87] who use Word2Vec for representing RDF data choice 25 as a number of negative samples. Word2Vec model is implemented using open-source Java package called Deeplearning4j (DL4J) [94], which is a suit of tool to build an application that runs a deep learning on JVM [95].

2.9 Recommender System Evaluation

Evaluation of a RS might be challenging for variety of reasons. For instance, given the same amount of data, an algorithm's performance might oscillate depending on the assessment objective. Additionally, changing evaluation setting

might have an effect on the evaluation result [34]. However, the majority of research in the domain of RSs is conducted offline, using existing datasets and a framework for modeling user behavior is followed to estimate recommender performance metrics such as prediction accuracy. A more costly approach is a user study, in which a selected group of users is requested to complete series of activities on the system, often followed by questionnaires about their experience. Finally, large-scale trials on a real system can be conducted, which is an online experiments. These experiments examine the recommender system's performance on real users.

A pre-collected dataset of user rating items is used in an offline experiment. The behavior of user who engage with a RS will be simulated using this information. This approach assumes that the user behavior when the data was gathered will be comparable enough to user behavior when the RS is implemented. Offline experiments are attractive because they do not involve real-world contact and thus enable the researchers to evaluate a large number of potential algorithms for a minimal cost [26].

Because the offline evaluation purpose is to filter algorithms, the data utilized for the evaluation must be as near as to the data that a RS would encounter when deployed online. However, the evaluation process is constrained by the computing cost of an assessment methodology and must make trade-offs in order to conduct the experiment over huge datasets. On the other hand, if it is possible to access to timestamps, the evaluation process will simulate what the system would have predicted if it had been operating at the time the dataset was acquired. It is possible to sample a group of test users and a single test time, then conceal all items for each test user after the sampled test time. This simulate a scenario in which the RS is constructed at the time of the test. Another possibility is to ignore the time. All evaluation protocols discussed by Ricci, et al. [26] make assumptions

to concern the behavior of the users when design an evaluation mechanism and make a consideration of the specific application and the underline domain.

In many applications, the RS does not attempt to predict the ratings, but rather attempts to recommend items that the user may be interested in. Typically, in an offline assessment, there is a dataset of items used by different users. A test user is chosen, then hide some of his rated items, and ask the recommender to predict N items to the user. Then, for the recommended and hidden items, we have four alternative outcomes, as shown in Table 2.2.

Table 2.3 Possible outcomes for recommendation

	Recommended	**Not Recommended**
Used	True-Positive (TP)	False-Negative (FN)
Not used	False-Positive (FP)	True-Negative (TN)

There is assumption said that unused items would not have been utilized even if they had been suggested since the data isn't normally obtained using the proposed RS and the items are worthless to the user. This assumption might be incorrect, for example, if the list of unused items includes some interesting items that the user did not choose. For example, a user may not have utilized an item because he does not know it exists, but now that the item has been revealed by the recommendation, the user might choose to use it. The number of false positives is overestimated in this situation [26].

There are two protocols to evaluate Top-N recommendation: the first is **all unrated items** protocol that creates a list of recommendation items that are not rated by the user regardless of whether the items are in the test list or not. After that, implementing different kinds of measures to compare the recommended items with the test set. However, this protocol underestimates real recommendation

quality. The second protocol is **rated test-items** that consider only the items exist in the test set to recommend top-N list [4].

Top-N recommendation utilizes measures such as precision, recall, F-measure, and Mean Reciprocal Rank (MRR) to compare a ranked list of suggested items to those put aside in a user profile. Every value in the above table is counted in order to find the precision and recall using the following equations [26, 96]:

$$Precision(P) = \frac{TP}{TP+FP} \tag{2.16}$$

$$Recall(R) = \frac{TP}{TP+FN} \tag{2.17}$$

According to Fayyaz, et al. [37] "Precision indicates the fraction of relevant items among all the recommended items to a user, and recall represents a number of relevant recommended items to the total number of items that should be recommended".

We may usually assume a trade-off between these two quantities: although enabling larger suggestion list increases recall, it also reduces precision. Precision at N is the most helpful measure of interest in situations when the number of suggestions that may be offered to the user is predetermined [26].

In applications where each user receives a fixed number of recommendations, the precision and recall can be computed for each user at each recommendation list length N, and then average the precision and recall over all users. The generated curves are very useful since they provide N numbers for each feasible precision and recall, and may be used to evaluate the performance for a given N. Receiver Operating Characteristic (ROC) is one that is produced in this way. ROC curves have been widely used to assess the effectiveness of RSs when evaluating the performance of different algorithms. Zhang, et al. [97] utilize the value of the area under the ROC curve as their assessment metric. ROC curves are

also used by Banerjee and Ramanathan [98] to compare the performance of different models [26].

On the other hand, the precision-recall (PR) curve is also widely used by researchers to evaluate RSs. It depicts the tradeoff between precision and recall at various thresholds, with a high Area Under the (PR) curve (AUPR) indicating good precision and recall [99]. Precision-recall curves focus on the proportion of recommended items that are truly relevant, whereas ROC curves focus on the proportion of recommended items that are not relevant. Precision-recall curve is more suitable for evaluating RS in cases that need to highlight the proportion of recommended items that are relevant such as when the system recommend a movie to watch [100].

Another measure which is considered as a harmonic mean of precision and recall is called f-measure. It is useful, since precision and recall give various information that can complete one another when combined. This measure can be calculated as follows: [26, 37]

$$f - measure(F) = 2 \times \frac{Precision \times Recal}{Precision + Recall} \tag{2.18}$$

Precision, recall, and F-measure all consider the results as a set, disregarding the order in which the positive outcomes are presented inside a rank-ordered list. As a consequence, many researchers choose to employ the Mean Reciprocal Rank (MRR), which considers how early a relevant result appears among ranked results. MRR is computed as follows: [96]

$$MRR = \frac{\sum_{i=1}^{|U|} \frac{1}{rank_i}}{|U|} \tag{2.19}$$

Where $rank_i$ is the highest rank of relevant results for a specific user U_i and the denominator is the number of users.

Komeiha et al. [12] proposed LDS library, a java library for LOD-based similarity measures, includes different measures such as LDSD, TLDSD, PICSS, EPICS, and others. This library helps researchers to achieve fair comparisons between the different methods using same data and same evaluation environment.

CHAPTER THREE

The Proposed System

3.1 Introduction

The knowledge included in the LOD cloud helps in the solution of situations when few or no characteristics define the items to be recommended. Many characteristics related to a recommendation task may be extracted from the LOD cloud to improve the quality of suggested items and meet the expectations of the user and consequently solving the problems of traditional RSs such as; cold-start, data sparsity, and the lack of semantics problems. The overall objectives of this dissertation are to design a linked data-based RS for multi-domain and present an algorithm that relies on linked data and makes use of the transversal relations in the LinkedMDB and DBpedia datasets, as well as the category structure in DBpedia, to locate potential resources for recommendations. Also, this work recommends items from different domains that the user does not expect to see. This chapter presents the stages of building the proposed recommender system in details and an explanation of the different techniques used to achieve our objectives.

3.2 The Proposed System Description

The proposed system is composed of seven stages as follows and as illustrated in Figure 3.1

1- Data preprocessing
2- Feature extraction
3- Items representation (Vectorization)
4- User representation
5- Movies recommendation
6- Recommending from other domains
7- Evaluation

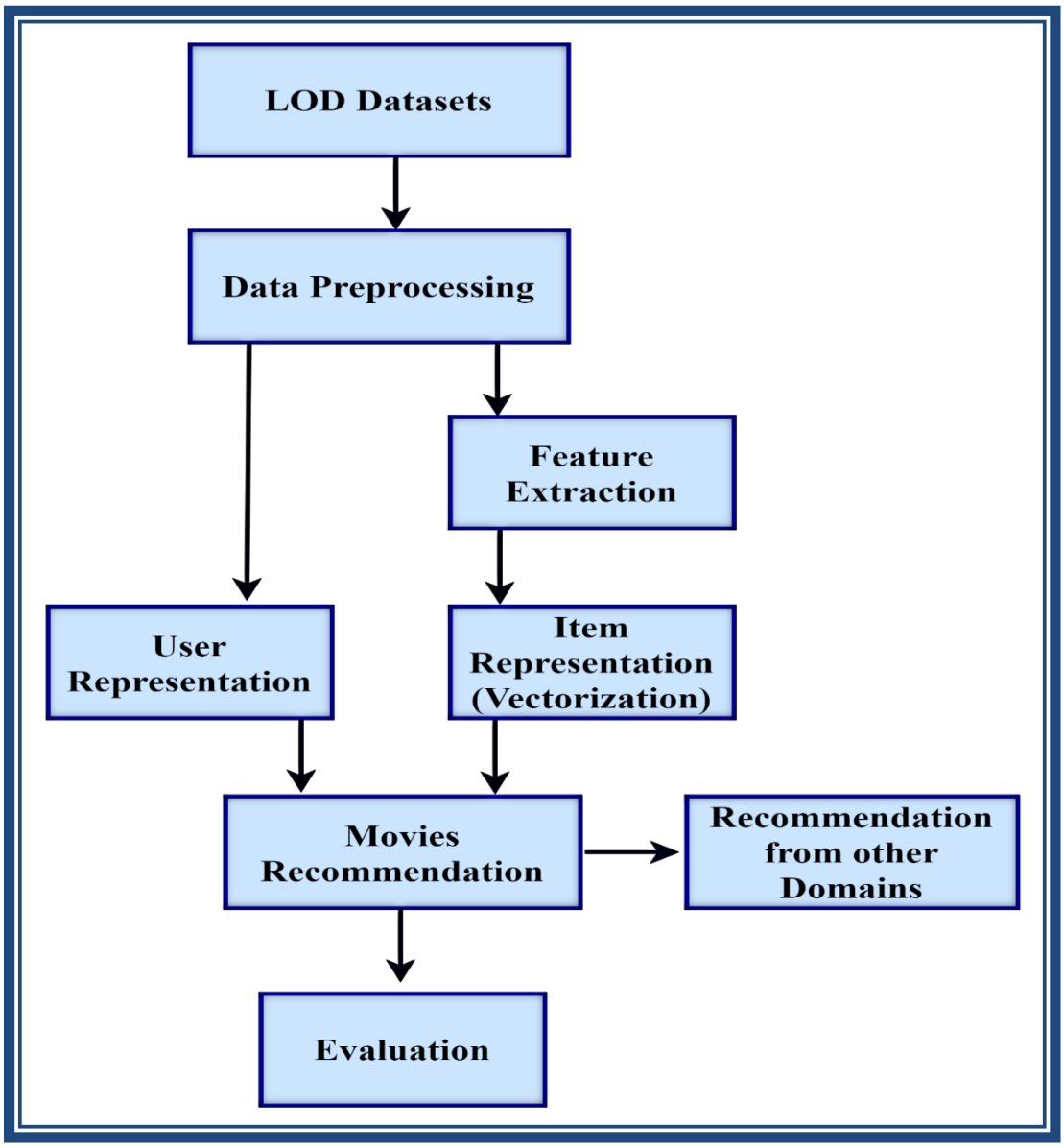

Figure 3.1. The Block Diagram of the Proposed System

3.3 LOD Datasets

The first step in the proposed approach is to select datasets from LOD cloud in order to recommend the related items. LOD cloud includes more than one thousand datasets in different domains. We choose the LinkedMDB dataset for the movies domain and DBpedia as a cross-domain dataset containing rich information about the movies. Furthermore, to recommend related Books of the Top-N movies,

there are some links in LinkedMDB called “relatedBook” which links a movie to a related book or novel in Book Mashup dataset. However, this dataset is deprecated and cannot be used anymore. Therefore, BNB Library dataset is used, which consists of metadata about books and different kinds of publications, to enrich the LinkedMDB with new links to BNB dataset in order to recommend the related books from this dataset.

To access the LinkedMDB, a dump file of this dataset is uploaded to local Virtuoso server in order to be accessible through a local endpoint. On the other hand, DBpedia and BNB Library datasets have huge dump files, therefore we depend on remote access to their endpoints. Table 3.1 below shows the endpoint URI for each selected datasets.

Table 3.1. Datasets endpoints

Dataset	Endpoint
LinkedMDB	http://localhost:8890/sparql
DBpedia	http://dbpedia.org/sparql [75]
BNB Library	https://bnb.data.bl.uk/sparql[101]

3.4 Data Preprocessing

The movies in DBpedia and LinkedMDB datasets are interlinked with *SameAs* relations, but they also need a preprocessing stage, which is shown in Figure 3.2, to manipulate some cases when some movies are without interlinking relations.

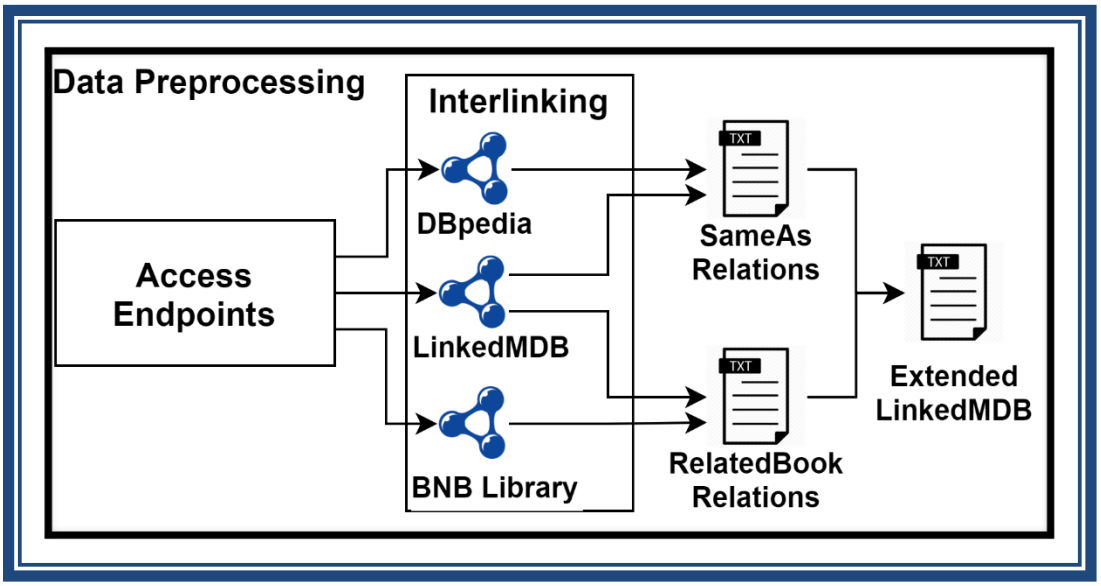

Figure 3.2. Data Preprocessing Stage

For example, there is no *SameAs* links between the ***"Die Hard"*** movie in DBpedia and its corresponding in LinkedMDB, therefore, to fix this missing and specifically, to answer the **RQ1** (mentioned in section 1.1) **,** Silk framework is used to enrich the LinkedMDB with more *sameAs* links to DBpedia. Silk framework needs some configuration in order to do the interlinking process such as; the source and the target endpoints URIs, in addition to the type of the resources that will be interlinked. If the movie's labels in the two datasets are exactly matched, the URI of the movie in LinkedMDB will be linked to the corresponding URI of the same movie in DBpedia through s*ameAs* relation and the statement generated will be stored in a separate file.

The number of *sameAs* links to DBpedia is originally 30354. However, after the interlinking process the links become 40142 which means an increase of about 9788 *sameAs* links from LinkedMDB movies to corresponding DBpedia movies. The generated file which includes all the generated *sameAs* links will be uploaded to the local Virtuoso Server to enrich the LinkedMDB which is already uploaded to the Server. The RDF statement below is an example of link generated for the

"*Die Hard"* movie to link the URI of the movie in LinkedMDB to its corresponding URI in DBpedia.

```
<https://triplydb.com/Triply/linkedmdb/id/film/38525>
<http://www.w3.org/2002/07/owl#sameAs>
<http://dbpedia.org/resource/Die_Hard> .
```

To solve **RQ2** (mentioned in section 1.1) and to fix some links in LinkedMDB that are connected by *"relatedBook"* to a deprecated dataset, extra links are added to LinkedMDB to link a movie to its related book in BNB Library through same property *"relatedBook"*. The link is added, if the writer of the movie and the name of the movie are matched with the writer of the book and the name of the book as shown below in Figure 3.3.

```
SELECT distinct ?s ?o ?list
WHERE {
  ?s <http://schema.org/name> ?o;
      rdf:type <http://purl.org/ontology/bibo/Book>;
      <http://schema.org/author>|<http://schema.org/contributor> ?x.
   ?x <http://xmlns.com/foaf/0.1/name> ?list.
FILTER(?list in (writerList))
FILTER(lcase(str(?o)) = lcase(movieName))
}
```

Figure 3.3. SPARQL query to extract book information from BNB Library

3.5 Feature Extraction

The purpose of this stage, which is shown in Figure 3.4, is to extract the triples existing in DBpedia and LinkedMDB datasets that denote to all resources

linked from a movie via specific properties so that the movies can be represented by their linked resources which are considered as features of the movie.

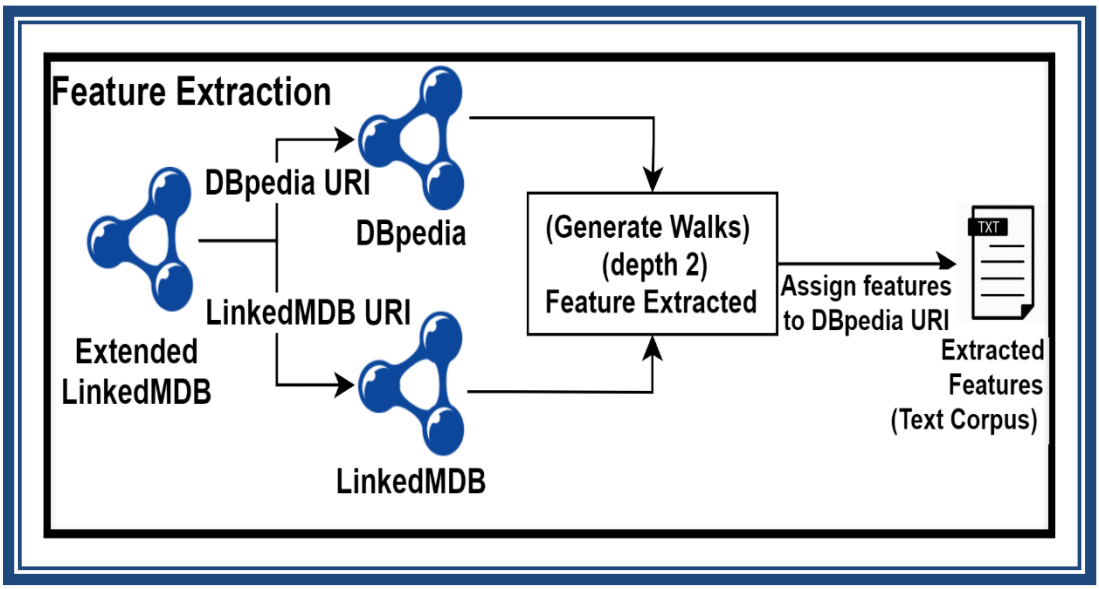

Figure 3.4. Feature Extraction Stage

To get the features of the movies from DBpedia and LinkedMDB, first, a SPARQL query is implemented on Extended LinkedMDB to get the URI of the movie from both datasets. After that, walks of depth 2 are generated from every movie URI by implementing another query to extract all the resources linked from the movie URI by specific properties on DBpedia as shown in Figure 3.5 and on LinkedMDB as shown in Figure 3.6.

```
SPARQL1= ”SELECT distinct ?s ?p ?o
WHERE {
    Values (?s ?p) {
      (<URI> <http://purl.org/dc/terms/subject>)
      (<URI> <http://dbpedia.org/ontology/director>)
      (<URI> <http://dbpedia.org/ontology/starring>)
      (<URI> <http://dbpedia.org/ontology/writer>)
    }
    {?s ?p ?o.}
}”
//URI is a variable includes the URI of the movie
```

Figure 3.5. Feature extraction from DBpedia

```
SPARQL2= “SELECT distinct ?s ?p ?o
WHERE {
  Values (?s ?p) {
   (<URI> <https://triplydb.com/Triply/linkedmdb/vocab/genre>)
   (<URI>  <https://triplydb.com/Triply/linkedmdb/vocab/director>)
   (<URI> <https://triplydb.com/Triply/linkedmdb/vocab/actor>)
   (<URI> <https://triplydb.com/Triply/linkedmdb/vocab/writer>)
  }
{?s ?p ?o.}
}”
//URI is a variable includes the URI of the movie
```

Figure 3.6. Feature extraction from LinkedMDB

The features of all movies are stored in a separate file, so that each movie will be represented by multi lines and each line consists of the DBpedia URI of the movie and a feature from DBpedia or LinkedMDB dataset; the features include actors, genre, writer and director URIs which are the most important interests to

each user who search for a movie. The Figure 3.7 illustrates a sample of features that can be extracted from the both interlinked datasets.

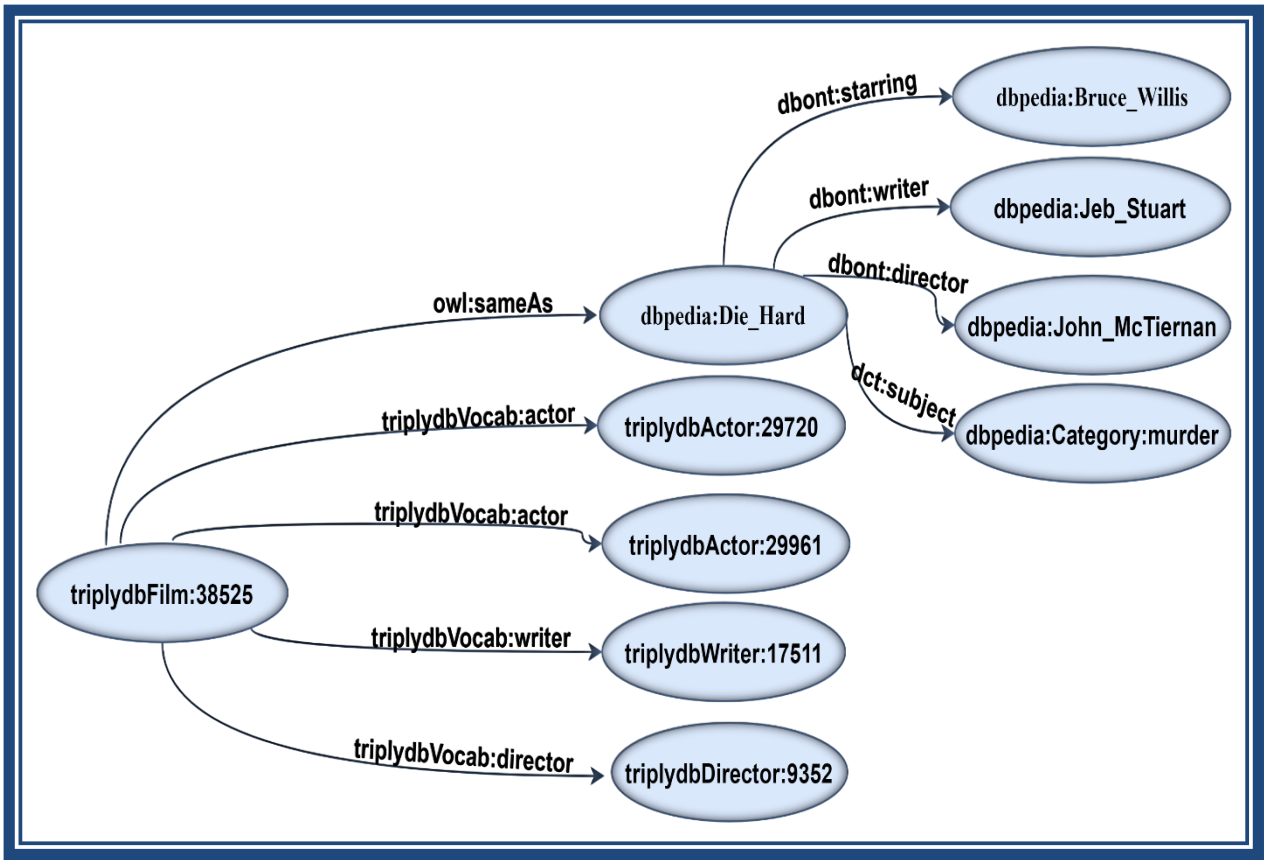

Figure 3.7. Sample of the "Die Hard" movie's features from two datasets

The whole procedure of extracting the features of the movie items illustrated in Algorithm 3.1.

Algorithm 3.1: Feature_Extraction

Input: *mdbURIs*: LinkedMDB movies' URIs
pediaURIs: DBpedia movies' URIs
Output: features: text file
BEGIN
Step1: CALL SPARQL1 for each pediaURI //SPARQL1 is a query in Figure 3.5
Step2: STORE a statement includes
pediaURI and the retrieved predicate and object to features file
Step3: CALL SPARQL2 for each mdbURI //SPARQL2 is a query in Figure 3.6
Step4: STORE a statement includes
pediaURI and the retrieved predicate and object to features file
Step5: Return features
END

Algorithm3.1 includes two lists of URIs as input, the first is *mdbURIs* that contains LinkedMDB URIs, while the second list is pediaURIs that contains the DBpedia URIs. Algorithm1 also includes the executing of two queries; SPARQL1 on DBpedia in step1 and SPARQL2 on LinkedMDB in step3. SPARQL1 and SPARQL2 are queries shown in Figure 3.5 and 3.6, respectively. The query on DBpedia and LinkedMDB will result a list of predicate and object that will be stored on the features file (step2 and step4) also as predicate and object, while the subject will be a DBpedia URI to make for every DBpedia URI, a list of features from both datasets as illustrated in Figure 3.8.

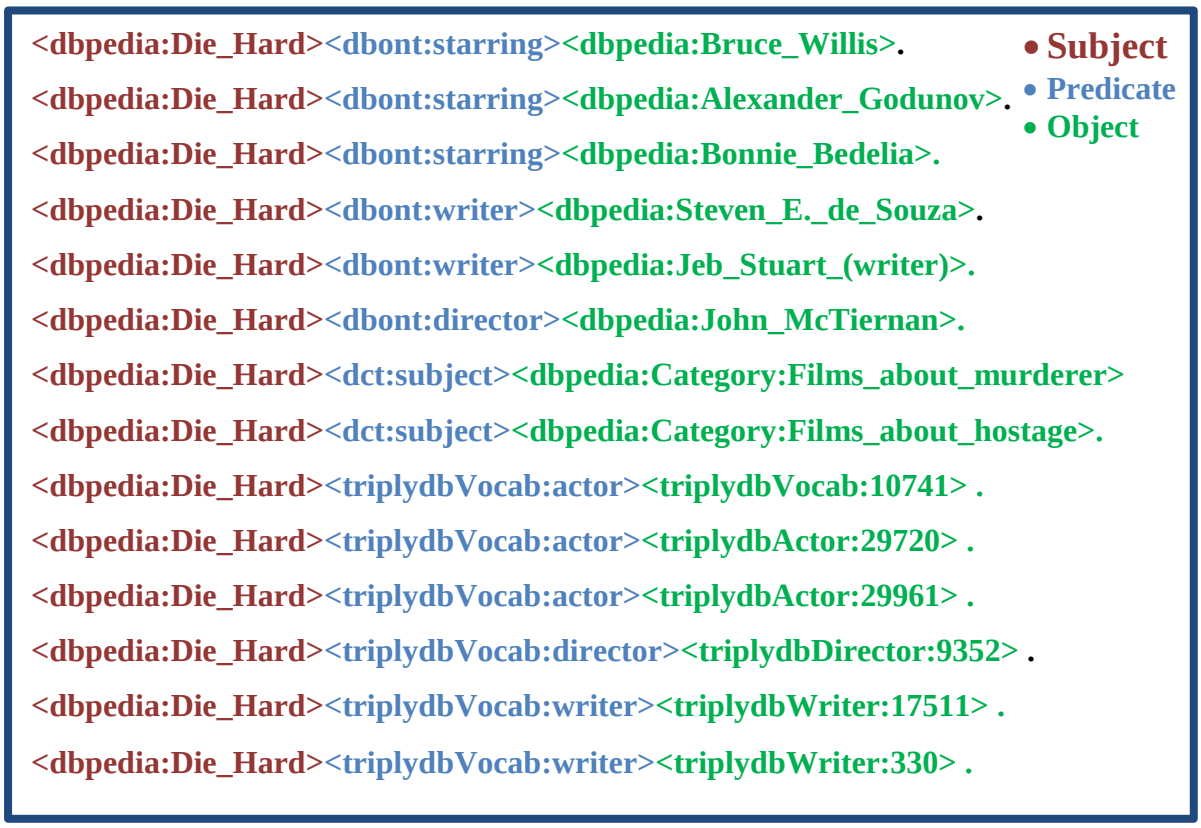

Figure 3.8. Sample of the data in the generated text file (features.txt)

As seen in the above Figure 3.8, the subject of every statement is a URI from DBpedia, while predicate can be a URI from DBpedia such as *dbont:starring* or from LinkedMDB such *triplydbVocab:actor*.

The generated list of statements for each movie will be used in the next section for Word2Vec model in order to represent the movies numerically by vectorizing their features.

3.6 Item Representation (Vectorization)

The features of the resources that are extracted in the previous stage must be represented numerically in vectorization stage (see Figure 3.9) in order to implement some operations to find the similarities.

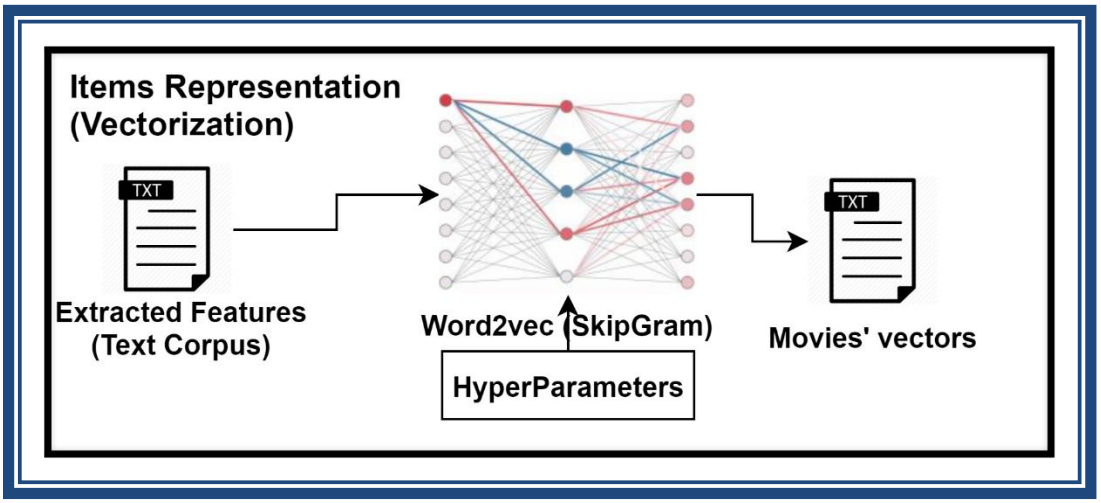

Figure 3.9. Item Representation Stage

In order to answer **RQ3** (mentioned in section 1.1) and specifically, the item representation part of this question, the Word2Vec model is utilized, which uses neural network to learn the word association. The vocabulary of the generated file (the extracted features) has 188212 unique words and the Word2Vec model will generate (target – context) pairs. Then, a One-Hot Encoding of the word will be fed to a neural network and multiplied with the hidden layer weights. The weights will be updated for training and finally, the vectors of the updated weights will be stored in a separate file to be used later for movie similarity calculation in order to rank and recommend the top-N movies.

3.7 User Representation

To answer the other part of **RQ3**, which is how to represent the user's profile, User Representation stage is presented in Figure 3.10. The first step in this stage is to extract the features of the user's history and create a file of interests for that user. The second step is building a user profile based on FOAF vocabularies and using the features extracted in the first step. Therefore, we will get a profile that can be used in RS task, and can be queried in efficient way using semantic query language (SPARQL).

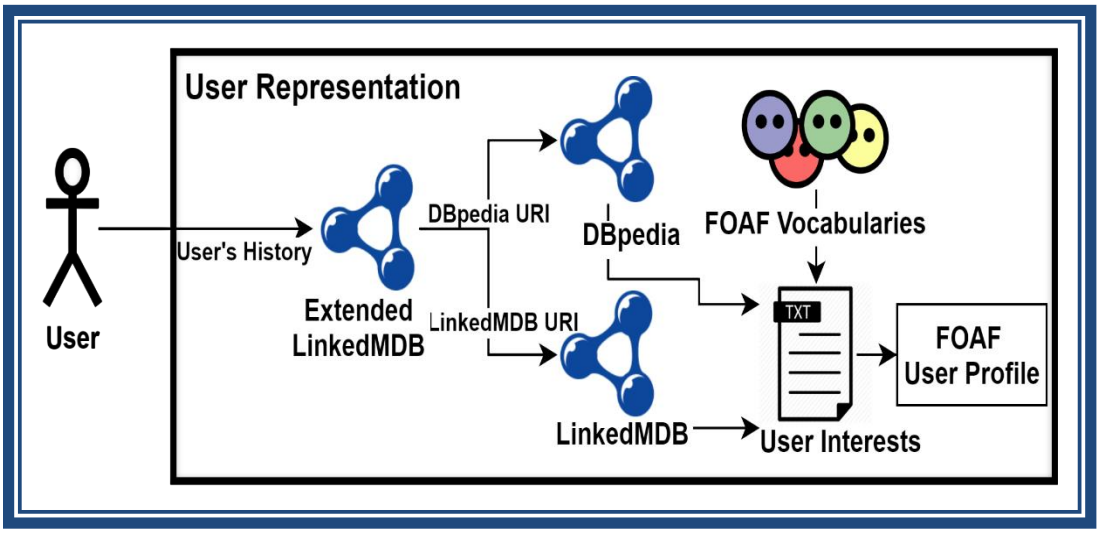

Figure 3.10. User Representation Stage

3.7.1 Extracting the Features of Interest

The user usually search, watch, and rate some movies and this history of user's behavior need to be analyzed to extract the features of the movies included in the history. For example, the user is interested in "The Terminator" movie, this movie has a DBpedia URI as follow:

<http://dbpedia.org/resource/The_Terminator>

In this case, it is needed to get the corresponding LinkedMDB URI in order to extract the features from both datasets. Therefore, a SPARQL query illustrated in Figure 3.11 will be sent to Extended LinkedMDB dataset to obtain the corresponding LinkedMDB URIs for all DBpedia URIs.

```
String sameas_query =
"SELECT distinct ?s ?o
 WHERE {
       ?s <http://www.w3.org/2002/07/owl#sameAs> ?o.
        FILTER regex(str(?o), "dbpedia").
        FILTER regex(str(?s), "https://triplydb.com/Triply/linkedmdb/id/film").
} "
```

Figure 3.11. A query to extract sameAs relations

Examples of *The Terminator* movie features are illustrated in Table 3.2.

Table 3.2. Features extracted from LinkedMDB and DBpedia

Feature Name	URI	From Dataset
Actor /Starring	https://triplydb.com/Triply/linkedmdb/id/actor/13309	LinkedMDB
	http://dbpedia.org/resource/Linda_Hamilton	DBpedia
Director	https://triplydb.com/Triply/linkedmdb/id/director/5617	LinkedMDB
	http://dbpedia.org/resource/James_Cameron	DBpedia
Genre /Subject	https://triplydb.com/Triply/linkedmdb/id/film_genre/23	LinkedMDB
	http://dbpedia.org/resource/Category:American_action_films	DBpedia
Writer	https://triplydb.com/Triply/linkedmdb/id/writer/8820	LinkedMDB
	http://dbpedia.org/resource/James_Cameron	DBpedia

Those features will be stored in a structured way in order to facilitate the process of accessing and using them later for the personalization of the recommendation task as will be explained in the next subsection.

3.7.2 Building a FOAF- based user profile

FOAF terms are reused to build a machine-readable user profile. *Homepage* property, a social web term in FOAF, is used and populated with the URIs of the movies seen in the history of the user. In addition to *interest* property, which is also considered as a social web term in FOAF vocabularies, in order to populate it with the features extracted in the previous step which represent the movies' features that the user is interested in. Therefore, the output of this step will be an RDF file that can be used by the RS and query it using the same known query language (SPARQL). The Figure 3.12 illustrates a graph visualization of this user profile, which uses the same features shown in Table 3.2.

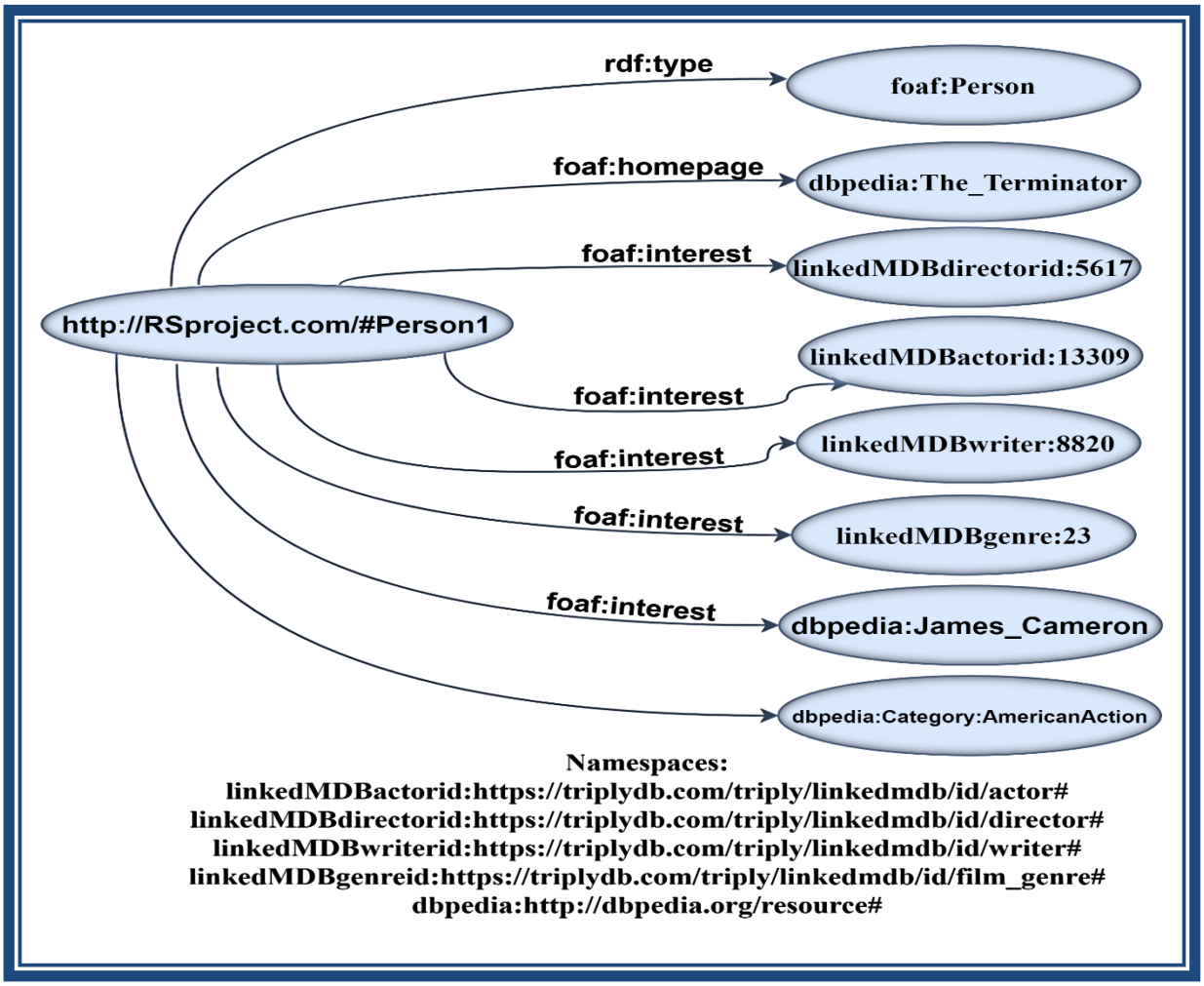

Figure 3.12. A graph representation of the FOAF-based user profile

Algorithm 3.2 below shows the steps of representing a user by extracting the features of interests and building the FOAF-based profile.

Algorithm 3.2: User Representation

INPUT: DBpedia_URIs: movies in user's history
OUTPUT: Profile: FOAF- based Profile (RDF)
BEGIN
Step1: CALL sameas_query //sameas_query is a query in Figure 3.11
Step2: CALL SPARQL1 for each *Object*(DBpedia URI) returned from Step1
//SPARQL1 is a query in Figure 3.10
Step3: CALL SPARQL2 for each *Subject*(LinkedMDB URI) returned from Step1
//SPARQL2 is a query in Figure 3.10
Step4: **ADD** results of Step2 and Step3 as values of *foaf:interst* property in *Profile*
Step5: **ADD** *DBpediaURIs* as values of *foaf*:*homepage* in *Profile*
Return Profile
END

3.8 Recommendation Process

The recommendation process has been divided into three steps: generation of candidates list, candidates ranking, and result presentation as illustrated in Figure 3.13.

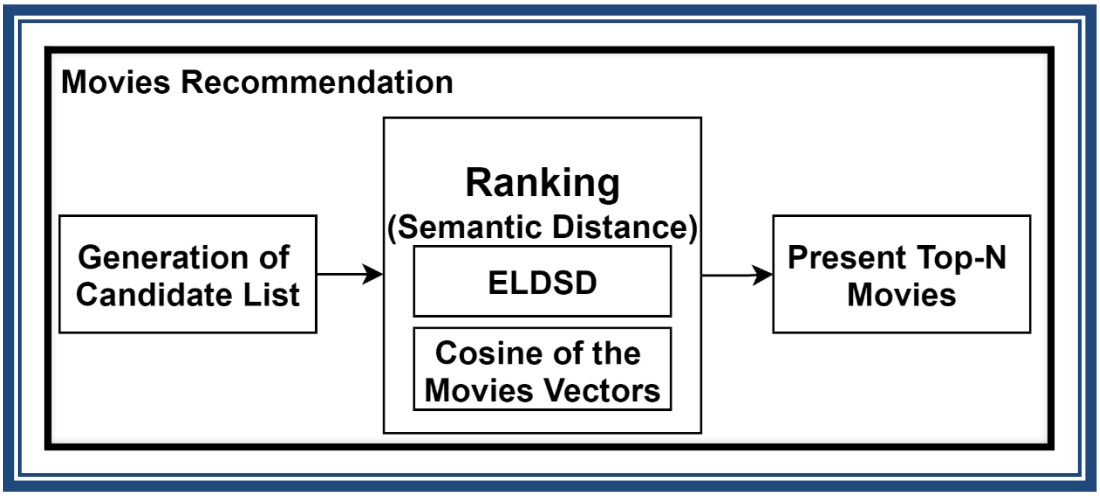

Figure 3.13. Movies Recommendation Stage

3.8.1 Generation of Candidates List

Generation of candidates list aims to discover the related resources to a given resource by analyzing the semantic relationships. The related resources will be generated with a condition that must be personalized by having the features exist in the user's profile. There are different kinds of relations in LOD, some of them are direct and others are indirect. In this section, an answer of the second research question **RQ4** (mentioned in section 1.1) will be answered by reviewing all kinds of relations exist and how to be utilized to produce recommendations. It is very important to analyze the direct relations that lead to most related items; for example, every Wikipedia page has hyperlinks to other pages embedded in the text of the page and there are a DBpedia ontology property to denote this type of relations between the pages known as *dbont:wikiPageWikiLink* as shown in Table

3.3. This kind of direct relations that connects a movie to the most related movies are utilized in the current approach.

Table 3.3. Examples of direct relations in DBpedia

?s	?p	?o
dbpedia:The_Terminator	dbont:wikiPageWikiLink	dbpedia:Terminator_2:_Judgment_Day

The direct link illustrated in the above Table 3.3 can be retrieved through a SPARQL query illustrated below in Figure 3.14. As can be seen, the query will extract any resource of the type "Film" that is connected from the Terminator Film directly by any predicate.

```
directQuery="PREFIX dbont: <http://dbpedia.org/ontology/>"
SELECT distinct ?o
WHERE {
  dbpedia:The_Terminator ?p ?o.
  ?o <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> dbont:Film.
}"
```

Figure 3.14. SPARQL query to extract the direct linked movies

The direct links between the movies also exist in LinkedMDB dataset but in different way. For example, we can find that two movies are linked to the same genre, actors, director, and writer. These four properties are useful, and the user maybe like the movies that the same preferred actors act. As shown in Table 3.4, the two movies "triplydbFilm:36614" and "triplydbFilm:33" that represent "The Terminator" and "The Terminator 2" respectively, both of them are linked through "*triplydbVocab:writer"* property to same writer "*writerid:9447"* (James Cameron). Another example in the same table shows a direct link as previously seen in DBpedia by linking "triplydbFilm:578" and "triplydbFilm:21316" through

"*triplydbVocab:sequel"* which means the first movie "*Terminator Salvation"* is a sequel of the second "*Terminator 3"*.

Table 3.4. Examples of direct relations in LinkedMDB

?s	?p	?conjunction	?o
triplydbFilm:36614	triplydbVocab:writer	writerid:9447	triplydbFilm:33
triplydbFilm:578	triplydbVocab:sequel		triplydbFilm:21316

Those types of relations are also clarified visually as graph in Figure 3.15

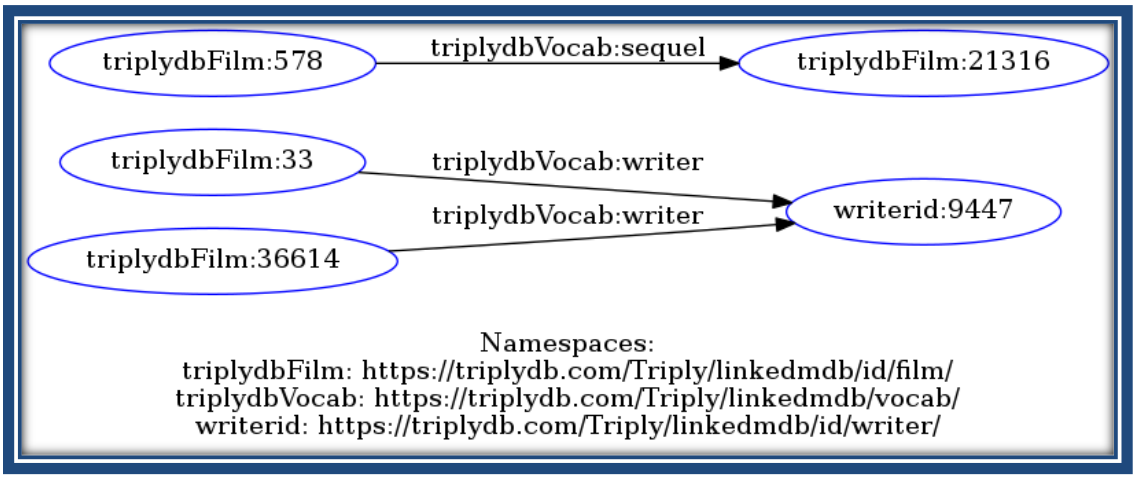

Figure 3.15. Direct relations in LinkedMDB

For the indirect links, as can be seen in Table 3.5, "The Terminator" movie has a link to "Adam Greenberg" who is a cinematographer and there is another link through *dbont:knownFor* property from "Adam Greenberg" to "Terminator 2" because he worked on this movie. This indirect link infers a similarity between the two movies so "terminator 2" can be a candidate for the recommendation.

Table 3.5. Examples of indirect relations in DBpedia

?s	?p1	?o1	?p2	?o2
dbpedia:The_Terminator	dbont:wikiPageWikiLink	dbpedia:Adam_Greenberg_(cinematographer)	dbont:knownFor	dbpedia:Terminator_2:_Judgment_Day

dbpedia:The_ Terminator	dbont:writer	dbpedia:James_ Cameron	dbont:wikiPage WikiLink	dbpedia:Terminator_2:_Judgment_ Day

The graph representation of the above indirect relations are shown below in Figure 3.16.

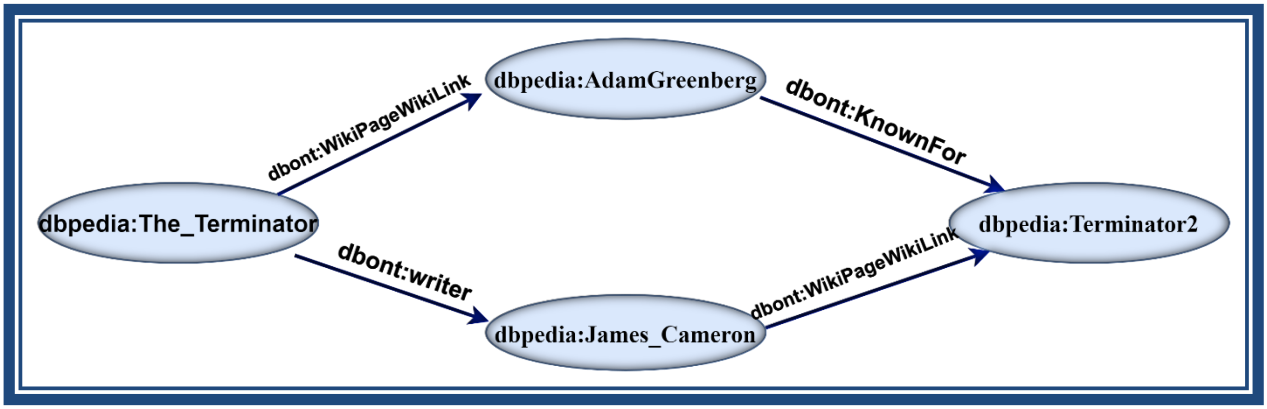

Figure 3.16. Indirect relations between the movies in DBpedia

The indirect relation in LinkedMDB is illustrated in Table 3.6 and Figure 3.17, as can be seen the movie "triplydbFilm:36614" (The Terminator) has a director **"**directorid:5617" (James Cameron) who made another movie "triplydbFilm:33" (Terminator2). Therefore, the two movies are somewhat similar and "Terminator2" will be a candidate.

Table 3.6. Examples of indirect relations in LinkedMDB

?s	?p1	?o1	?p2	?o2
triplydbFilm :36614	triplydbVocab: director	directorid:5617	foaf:made	triplydbFilm:33

Figure 3.17. Indirect relations between movies in LinkedMDB

The categories of resources in DBpedia can be obtained by *dcterms:subject* relation and the upper level of this category can be obtained through skos:broader relation as illustrated in Table 3.7. For example, "The Terminator" movie has a category called "Category:Films_directed_by_James_Cameron", this category includes all movies directed by "James Cameron". The upper level of this category called "Category:Films_by_Canadian_directors" which includes all movies directed by Canadian directors and "James Cameron" among them, so other movies which may be in different genre but the director is of Canadian origin can be candidates for recommendation. The same thing in the second example which links the terminator to the category "Terminator_(franchise)_films" and upper category "Category: American_robot_films ". Therefore we can recommend to users not just movies under Terminator category but any American Robotics Films. For example, *"I_Robot"* movie can be recommended because it has a link to American robot films category which is a broader category of *"The_Terminator"* movie as shown below in Figure 3.18.

Table 3.7. Examples of direct and indirect categories

?s	dcterms:subject	skos:broader
Dbpedia: The_Terminator	Dbpedia: Category:Films_directed_by_James_Cameron	Dbpedia: Category:Films_by_Canadian_directors

Dbpedia: The_Terminator	Dbpedia: Category:Terminator_(franchise)_films	Dbpedia: Category:American_robot_films

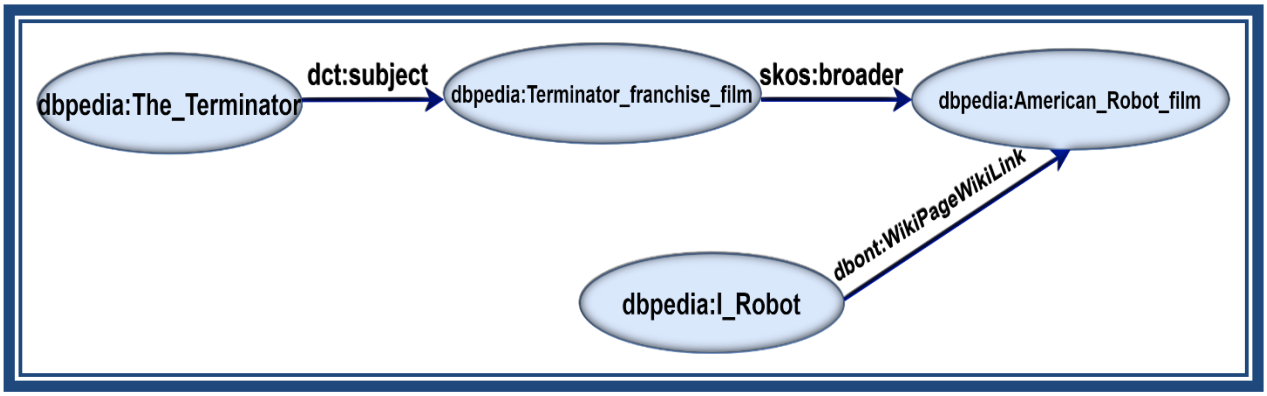

Figure 3.18. Example of indirectly linked movie through broader category

In this case, all direct and indirect categories can be extracted from all movies found in the history to extract all movies that share the same categories and consider them as candidates for recommendation. This step can be done through a SPARQL query illustrated in Figure 3.19.

```
String categoryQuery =
"PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dcterms:<http://purl.org/dc/terms/>
 PREFIX skos:<http://www.w3.org/2004/02/skos/core#>
 SELECT  ?subject
 WHERE {
                { <URI> dcterms:subject ?subject.}
                UNION
                {<URI>  dcterms:subject ?subject2.
                 ?subject2 skos:broader ?subject. }
 } ;"
```

Figure 3.19. A SPARQL query to get the direct and indirect categories of a movie

The prefixes mentioned in Tables 3.3-3.7 represent specific namespaces as defined in Table 3.8 :

Table 3.8. Prefixes and their namespaces

Prefix	namespace
triplydbFilm	**“**https://triplydb.com/Triply/linkedmdb/id/film/”
triplydbVocab	**“**https://triplydb.com/Triply/linkedmdb/vocab/”
writerid	“https://triplydb.com/Triply/linkedmdb/id/writer/”
directorid	“https://triplydb.com/Triply/linkedmdb/id/director”
foaf	“http://xmlns.com/foaf/0.1/”
dbpedia	**“**http://dbpedia.org/resource/”
dbont	“http://dbpedia.org/ontology”
Dcterms	“http://purl.org/dc/terms/”
skos	“http://www.w3.org/2004/02/skos/core#”

In DBpedia, we consider a link that is in one hop distance because many links of this kind of relations in DBpedia while in LinkedMDB there is few such links. Instead, there are direct links by sharing the same feature (actor, director, writer, or genre).

According to the above discussion about the types of links existing in LOD, three types of recommendations are proposed and summarized as follows:

1- Recommendation of the items directly linked to the resources seen in history
2- Recommendation of items that have same actor, genre, director, and writer.
3- Recommendation of items that have same direct and indirect categories.

Algorithm 3.3 shows how to recommend movies using only the direct links. In this algorithm, the Profile file, which is an input in this algorithm, must be read

to extract the resources that are linked by ***foaf:HomePage*** property (step1). Then, a SPARQL query, that called ***directQuery*** in Figure 3.13, will be executed on DBpedia dataset to get the related direct movies that are linked directly to the resources extracted from Profile (see step2). After that, a similarity measure will be implemented to get the score of similarity between the movies seen in history returned by step1 and the movies retrieved in step2. Finally, the candidates will be sorted according to the similarity score to present the top N movies (step4).

Algorithm 3.3: GetDirectFilms
Input: Profile //FOAF-based user profile
Output: Most related movies (URI_1, URI_2, URI_3,... URI_n)
BEGIN
Step1: READ Profile to get movies linked by foaf:HomePage (history of user)
Step2: CALL directQuery for each movies returned from Step1
// A query in Figure 3.11 to get directly linked movies
Step3: **Calculate** the semantic distance score between movies returned from Step1 and Step2
Step4: Sort movies returned from Step2 according to scores calculated in Step3
Return Top N movies
END

Algorithm 3.4 shows the recommendation process of items that have the same feature interested by the user. In step1, a SPARQL query is executed to read the file and get the features interested by the user from the user's Profile. After that, the features will be included in another SPARQL query that select all the movies that have the same interested features (step2). Similarity score also will be calculated and the candidate will be sorted to present the top N (step 3 and step4).

Algorithm 3.4: GetFilms_with_same_Feature

Input: Profile //FOAF-based user profile
Output: Most related movies (URI_1, URI_2, URI_3,... URI_n)
BEGIN
Step1: READ Profile to get values of foaf:interset properties
Step2: CALL SPARQL query to get movies with same interest features returned from Step 1 //same actor, director, writer and genre
Step3: **Calculate** the semantic distance score between movies returned from Step1 and Step2
Step4: Sort movies returned from Step2 according to scores calculated in Step3
Return Top N movies
END

In Algorithm 3.5, a SPARQL query will be executed on DBpedia and the hierarchical relations will be exploited to get the related movies that share the same direct and indirect categories. Firstly, as written in (step1 and step2) read the Profile and extract the categories of the movies found in this Profile by categoryQuery, a SPARQL query shown in Figure 3.18. Secondly, another SPARQL query will be executed to get all movies with the same interested categories (step3). Finally, as the previous algorithm, the similarity will be calculated and presenting the top N movies (step4 and step5).

Algorithm3.5 : GetFilms_with_same_Category

Input: Profile //FOAF-based user profile
Output: Most related movies (URI_1, URI_2, URI_3,... URI_n)
BEGIN
Step1: READ Profile to get movies linked by foaf:HomePage (history of user)
Step2: CALL categoryQuery for each movies returned from Step1 // A query in Figure 3.11 to get the categories of history
Step3: CALL SPARQL query to get movies with the same categories returned from Step2
Step4: Calculate the semantic distance score between movies returned from Step1 and Step3
Step5: Sort movies returned from Step3 according to scores calculated in Step4
Return Top N movies
END

3.8.2 Ranking and Presenting Top-N

The generated candidates list needs to be ranked in order to present the Top-N movies to the user. Therefore, a semantic distance between each candidate and the movies found in the history of the user need to be calculated. To calculate the semantic distance between the resources, two methods have been proposed; the first is depending on the direct and indirect links between the resources and to achieve this, a variant of LDSD called ELDSD is proposed. The second method depends on the features of the resources thus Word2Vec model is utilized and the resulted trained vectors are used to find the similarity of the resources through cosine distance of the vectors.

ELDSD is a re-implementation of LDSD, which is explained in section 2.7.1 and its equation illustrated in equation (2.1). In this proposed measure, the effect of the direct links is increased by multiplying the number of direct links by a factor K as shown in equation (3.1). The range of this measure is between 0 and 1.

$$\boldsymbol{ELDSD}(\boldsymbol{r_a}, \boldsymbol{r_b}) = \frac{\mathbf{1}}{\mathbf{1}+(\boldsymbol{K}\times\boldsymbol{D})+\boldsymbol{I}} \quad , \boldsymbol{K} \in \boldsymbol{R},\ \boldsymbol{K} > \mathbf{1} \tag{3.1}$$

The direct links (D) and indirect links (I) are same as equations (2.2) and (2.3), respectively, which are illustrated in Section 2.7.1. To calculate ELDSD score, a SPARQL query on the datasets is needed to extract the number of direct

```
SELECT ?s  (COUNT(?p) as ?indirect) (COUNT(?p2) as ?direct)
WHERE { VALUES (?s ?o) {(URI1 URI2)}
    {?s ?pp ?conj. ?conj ?p ?o.  } //indirect links
    UNION
    {?o ?p ?conj. ?conj ?pp ?s.}//indirect links (the inverse direction)
    UNION
    {?s ?p2 ?o} UNION {?o ?p2 ?s}//direct links in both directions
}
GROUP BY ?s
//URI1 is a candidate resource
//URI2 is a resource in the user's profile
```

Figure 3.20. Extract the number of direct and indirect relations from DBpedia

and indirect links between the resources for both direction (ingoing and outgoing) as shown in Figure 3.20 and Figure 3.21.

```
SELECT ?s  (COUNT(?p) as ?indirect) (COUNT(?p2) as ?direct)
WHERE { VALUES (?s ?o) {(URI1 URI2) }
  {?s ?p ?conj.   ?conj ?pp ?o }//indirect links
  UNION
  {?s ?p2 ?conj.  ?o ?p2 ?conj.  }//indirect links (the inverse direction)
  UNION
  {?s ?p2 ?o.} UNION {?o ?p2 ?s.} //direct links (in both directions)
}
GROUP BY ?s
```

Figure 3.21. Extract the number of direct and indirect relations from LinkedMDB

In the above Figures (3.20 and 3.21), the query will count all indirect links that are represented by a specific property (?p) and direct links that are represented by a property (?p2) according to the matching of the pattern shown in WHERE clause.

The value of K is chosen after several experiments to find the best value that enhance the final evaluation metrics used. The similarity between two resources can also be calculated by the cosine of the angle between the features of the two resources. This measure will be implemented on the data that is trained using Word2Vec algorithm.

ELDSD is useful in cases where the resources are well linked. However, sometimes the resources are poorly linked although there are features in common. Therefore, the proposed approach will decide to implement the second method that depend on the features and calculate the cosine distance of the vectorized features. By using this hybrid measure to rank the candidates and consequently recommend the Top-N movies to the user, we answered the **RQ5** (mentioned in Section1.1)**.**

Algorithm 3.6 illustrated the steps of the ranking and presenting the top-n movies in details.

Algorithm 3.6: Ranking and Presenting Top-N movies

Input: *history:* URIs of the movies found in history of users
Candidates: URIs of the candidate movies
trainedVectors: the vectors of movies obtained from Word2Vec model
threshold: a specific value that considers the resources are poorly linked if the similarity is exceed it

Output: Top N movies

BEGIN

1. **FOREACH** *candidate* **in** *Candidates* **do**
2. *score* ← ELDSD(*history*, *candidate*) //ELDSD is equation (3.1)
3. **IF** *score* **GreaterThan** *threshold*
4. **Extract** the vectors of movies found in *history* from *trainedVectors*
5. **Extract** the vector of each *candidate* from trainedVectors
6. *score* ← **Cosine** of the vectors returned from line4 and line5
7. **ENDIF**
8. **ENDFOR**
9. **Sort** Candidates according to *score* values
10. **RETURN** Top N movies

END

3.9 Recommending Resources from Other Domains

LOD datasets include different kinds of relations related to different domains so that the recommendation can recommend items from more than one domain. In addition, RS can exploit datasets from different domain to generate a list of various and unexpected items from different domains, but related in some way to the recommended Top-N movies. This stage is illustrated in Figure 3.22.

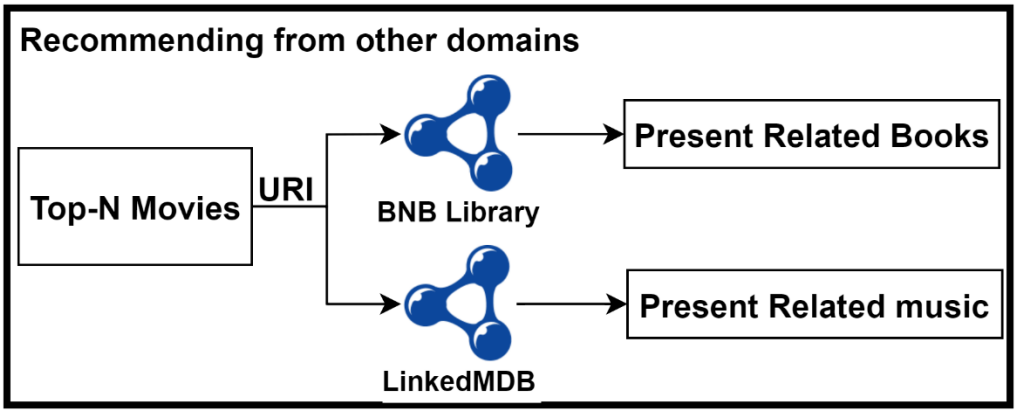

Figure 3.22. Recommendation Resources from other Domains

A SPARQL query on BNB library dataset needs to be implemented to collect the books/novels authored by the same movie's writer and have the same movie's name in order to recommend the related books to the user and this is done by the query illustrated in Figure 3.23.

```
Query1="SELECT distinct ?s ?o ?list
WHERE {
  ?s <http://schema.org/name> ?o;
     rdf:type <http://purl.org/ontology/bibo/Book>;
     <http://schema.org/author>|<http://schema.org/contributor> ?x.
  ?x <http://xmlns.com/foaf/0.1/name> ?list.
Filter(?list in (writersList))
Filter(lcase(str(?o)) = lcase(movieName))
}"
\\ writersList and movieName are variables contain the list of writer names and
   the list of movies' name
```

Figure 3.23. SPARQL for extracting related books

However, in the proposed system there is no need to query the related book in this way, since, the proposed Extended LinkedMDB is enriched with *"related_book"* relations to BNB Library as explained in Section 3.4, therefore, the related book can be queried directly from LinkedMDB using the query in Figure 3.24.

```
Query2="SELECT ?o ?name
WHERE {
   ?topnMovie <https://triplydb.com/Triply/linkedmdb/vocab/relatedBook> ?o.
   ?o <http://schema.org/name>  ?name
}"
```

Figure 3.24. Get the relatedBook from LinkedMDB directly

In addition, a property in LinkedMDB called *film_featured_song* property, connecting a movie with the songs used in the movie, is utilized to recommend the featured song to the user and the singer name by another property called *film_featured_song_performed_by* (see Figure. 3.25).

```
Query3="SELECT distinct ?s ?o ?label ?singer" +
WHERE {
?s <http://www.w3.org/2002/07/owl#sameAs> topnMovie;
   <https://triplydb.com/Triply/linkedmdb/vocab/film_featured_song> ?o.
   ?o rdfs:label ?label;
    <https://triplydb.com/Triply/linkedmdb/vocab/film_featured_song_performe
     d_by> ?singer
}"
//topnMovie is a DBpedia URI of the recommended movie
```

Figure 3.25. SPARQL for extracting song related to movie with the singer name

Algorithm 3.7 illustrates the procedure of getting books and songs by reading every movie in TOP-N movies and extracting its label and the writers from LinkedMDB. After that, this information is used to execute SPARQL *Query2* (shown in Figure 3.24) in order to get the recommended books with the name of

the writer and *Query3* (shown in Figure 3.25) in order to get the song name and the singer name.

Algorithm 3.7 : Recommend_Book_song

Input: TOP-N Movies (DBpedia URIs)

Output: related books,songs and singer

1. **BEGIN**
2. **FOREACH** movie in TOP-N
3. Extract the Label and writersName from LinkedMDB
4. Execute Query2 to get related book //Query2 in Figure 3.22
5. Execute Query3 to get related song and singer's name //Query3 in Figure 3.23
6. **ENDFOR**
7. **Return** Book name, writer name, Song name, singer name
8. **END**

The results will be as shown in the examples below:

Movie Name	**Related book**	**Writer name**
Natural born killers	Natural born killers	Quentin Tarantino
Gone with the wind	Gone with the wind	Margaret Mitchell

Movie Name	**Related song**	**Singer name**
Color of Night	The Color of the Night	Lauren Christy

CHAPTER FOUR

Implementation and Experimental Results

4.1 Introduction

This chapter presents the implementation of the proposed system step by step by explaining every stage in details and by examples to clarify the work as a whole. In addition to present the evaluation process and the results obtained to highlight the strength of this approach over other previous approaches.

The proposed system is developed using Eclipse environment and Java is used as a programming language. Virtuoso Server is used as a local endpoint to hold the LinkedMDB dataset. To extract data from the local and remote endpoints, Apache Jena API for Java is used and different SPARQL queries are executed to achieve that.

4.2 LOD Datasets

The first step in the proposed system is to select the datasets from the LOD cloud that can help to enrich the items with sufficient features and produce rich information to recommend items from different domains such as movie, music and books. Every dataset has either a dump file, which can be used in Local Virtuoso Server or a remote endpoint, which can be accessed remotely as discussed in section 3.3. The proposed system leverages three datasets: DBpedia, LinkedMDB, and BNB library. The position of the three datasets in the LOD cloud is illustrated in Figure 4.1. Those datasets need a preprocessing step that interlinks the related items by some kinds of relations in order to have features from multiple datasets and recommend items from different domains.

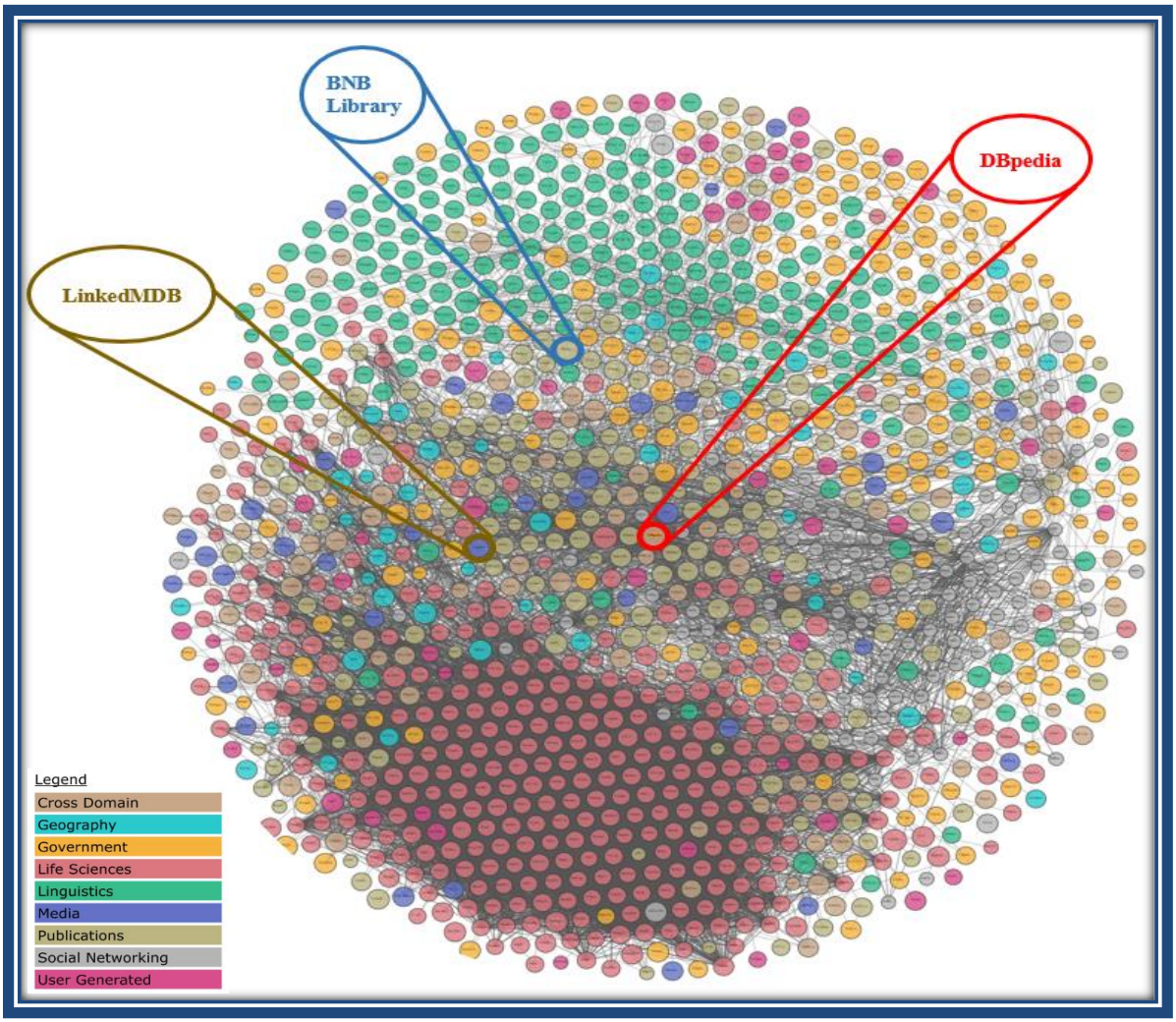

Figure 4.1. Position of the Three LOD Datasets

4.3 LOD Datasets Interlinking

There are two types of interlinking needed in those datasets. The first one is ***sameAs*** relation, which links every resource that represents a specific movie in LinkedMDB to the resource that represents the same movie in DBpedia, so that for every movie in LinkedMDB, we can extract its features from LinkedMDB and DBpedia at the same time. A sample of the generated ***sameAs*** links is shown in Figure 4.2.

1. "<https://triplydb.com/Triply/linkedmdb/id/film/110>
   <http://www.w3.org/2002/07/owl#sameAs>
   <http://dbpedia.org/resource/Alexander's_Ragtime_Band_(film)> ."
2. "<https://triplydb.com/Triply/linkedmdb/id/film/13680>
   <http://www.w3.org/2002/07/owl#sameAs>
   <http://dbpedia.org/resource/Shinjuku_Outlaw> ."
3. "<https://triplydb.com/Triply/linkedmdb/id/film/18404>
   <http://www.w3.org/2002/07/owl#sameAs>
   <http://dbpedia.org/resource/The_Ark_of_the_Sun_God> ."

   •
   •
   •

4. "<https://triplydb.com/Triply/linkedmdb/id/film/97233>
   <http://www.w3.org/2002/07/owl#sameAs>
   <http://dbpedia.org/resource/Man_Wanted> ."
5. "<https://triplydb.com/Triply/linkedmdb/id/film/12173>
   <http://www.w3.org/2002/07/owl#sameAs>
   <http://dbpedia.org/resource/The_Woman_and_the_Stranger> ."

Figure 4.2. A Sample of Generated SameAs Links

As noted in the above figure there are 5 statements as sample of the generated links and the numbers 1 to 5 refer to the different statements generated which include the URI of the movie from LinkedMDB, SameAs property, and the URI of the same movie in DBpedia.

The second relation needed is ***relatedBook***, which links a movie in LinkedMDB to a book or novel in BNB Library. This step is necessary to allow the proposed system to recommend the books that are related to the recommended movies. A sample of the results is shown in Figure 4.3. The Silk framework is used for the first type because it allows to add *SameAs* relation between the matched resources, while a java code is built for the second type as explained in section 3.4.

1. "<https://triplydb.com/Triply/linkedmdb/id/film/1985> <https://triplydb.com/Triply/linkedmdb/vocab/relatedBook> <http://bnb.data.bl.uk/id/resource/009709188>"
2. "<https://triplydb.com/Triply/linkedmdb/id/film/3681> <https://triplydb.com/Triply/linkedmdb/vocab/relatedBook> <http://bnb.data.bl.uk/id/resource/012666609>"
3. "<https://triplydb.com/Triply/linkedmdb/id/film/38078> <https://triplydb.com/Triply/linkedmdb/vocab/relatedBook> <http://bnb.data.bl.uk/id/resource/010978450>"

•
•
•

4. "<https://triplydb.com/Triply/linkedmdb/id/film/722> <https://triplydb.com/Triply/linkedmdb/vocab/relatedBook> <http://bnb.data.bl.uk/id/resource/018825897>"
5. "<https://triplydb.com/Triply/linkedmdb/id/film/8236> <https://triplydb.com/Triply/linkedmdb/vocab/relatedBook> <http://bnb.data.bl.uk/id/resource/010703827>"

- **Subject**
- **Predicate**
- **Object**

Figure 4.3. A Sample of Generated relatedBook Links

As noted in the above figure number 1 to 5 refer to five statement every statement includes LinkedMDB URI, ***relatedBook*** property, and the URI of the book in BNB library. As mentioned in Section 3.4, this type of relation is already exist in LinkedMDB and it links 700 movies in LinkedMDB to their related books in deprecated book dataset that is no longer accessible. However, the proposed Extended LinkedMDB succeed to match 680 movies in LinkedMDB to their related books in BNB Library. Therefore, there are only 20 missing matches.

The two types of the generated links are combined and added to the original LinkedMDB dataset to produce the Extended LinkedMDB dataset that is uploaded to Quad store using local Virtuoso server as explained in Section 3.4. The Quad store will be accessed later in user and items representation stages to extract the different kinds of features existing in the two datasets.

4.4 Enhanced Linked Data Semantic Distance

ELDSD measure adds more importance to the direct links over the indirect links by multiplying the number of direct links by K, where K∈ ℝ. Choosing of the best K value depends on trial and error by varying the K by different values from K=1 which refers to the original LDSD to K=15, evaluating the RS with MovieLens dataset, and calculating the precision, recall, and f-measure for each case . We found that the value of K that leads to highest P, R and F score is equal or greater than 3 as shown in Table 4.1 and presented in Figure 4.4.

Table 4.1. Evaluation of ELDSD with different values of K (MovieLens)

	K=1	K=2	K=3	K=4	K=5	K=10	K=15
P	0.56	0.56	**0.57**	0.57	0.57	0.57	0.57
R	0.311	0.311	**0.324**	0.324	0.324	0.324	0.324
F	0.399	0.399	**0.413**	0.413	0.413	0.413	0.413

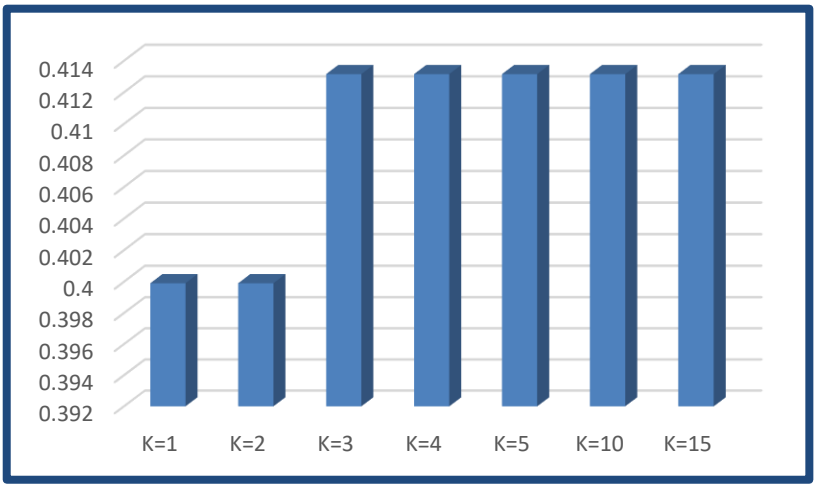

Figure 4.4. Comparison of F-measure on different K values in ELDSD (MovieLens)

Extra experiment is done with another dataset (Yahoo! Movie dataset) and the best K value, that leads to enhancement over the traditional LDSD, is when K

equals or is greater than 2 as shown below in Table 4.2 and presented as bar chart in Figure 4.5.

Table 4.2. Evaluation of ELDSD with different values of K (Yahoo!)

	K=1	K=2	K=3	K=4	K=5
P	0.585	**0.599**	0.599	0.599	0.599
R	0.320	**0.347**	0.347	0.347	0.347
F	0.414	**0.439**	0.439	0.439	0.439

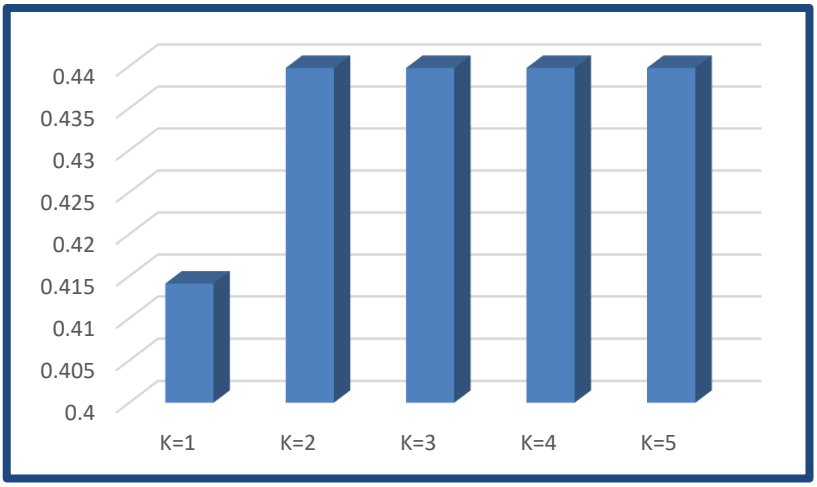

Figure 4.5. Comparison of F-measure on different K values in ELDSD (Yahoo!)

As shown in both experiments, in Tables 4.1 and 4.2, the value of K that can lead to better results for both datasets is when K=3 or greater, therefore, this value is chosen to be set in the proposed approach.

4.5 Case Study

To clarify the whole process of recommending items, we will start with a specific user (USER1) who has the history that its sample shown in Figure 4.6. This history represents the URIs of the movies seen by this user which are obtained from LinkedMDB and DBpedia.

“http://dbpedia.org/resource/Mars_Attacks!”
“http://dbpedia.org/resource/Maverick_(film)”
“http://dbpedia.org/resource/The_Crow_(1994_film)”
“https://triplydb.com/Triply/linkedmdb/id/film/76”
“http://dbpedia.org/resource/Batman_(1989_film)”
“http://dbpedia.org/resource/Honey,_I_Shrunk_the_Kids”
“http://dbpedia.org/resource/For_Your_Eyes_Only_(film)”
“https://triplydb.com/Triply/linkedmdb/id/film/33”
“https://triplydb.com/Triply/linkedmdb/id/film/1364”
“https://triplydb.com/Triply/linkedmdb/id/film/5514”

Figure 4.6. Sample of the User's History

4.5.1 Item Representation

The movies seen in the history of a user need to be represented numerically, therefore, the features of those movies will be extracted from LinkedMDB and DBpedia and stored in a file that includes movie’s URI and its features from both datasets. Figure 4.7 shows a sample of the features extracted for the movie “To End All Wars”.

Figure 4.7. Sample of the Extracted features of a movie

As noted in the above figure, the file generated consists of a set of words (Subject, Predicate, and Object) that make up sentences, therefore, Word2Vec is

used to vectorize each word and Deeplearning4J, which is discussed in section 2.8.1, is used to implement Word2Vec model. There are some hyper-parameters in Word2Vec that are configured as follow:

LayerSize (dimensions): The default number of dimensions of the embeddings is 100. However, this default value is often used when the training data is so big as in Google news; therefore, it can be lower for smaller training data and it depends on the task. After several experiments on different sizes (50,100, and 200), it is found that 50 is sufficient and is the best for this task with 0.381 f-measure as shown in Table 4.3.

Table 4.3. Experiments on different layer sizes

Layer Size	P	R	F
50	0.55	0.292	0.381
100	0.535	0.277	0.365
200	0.52	0.272	0.357

Window size: The default window size of the context is 5. However, this parameter is set to 2, since the training data in the proposed system includes statements with just three words, as we deal with RDF statements (Subject, Predicate, Object). To validate this choice, experiments are conducted and the window size is set to 2 and 5 to do a comparison. It is found, as expected, that 2 leads to better results with 0.381 f-measure as shown in Table 4.4. It is because 2 is the correct context for RDF statement which contains only three words.

Table 4.4. Experiments on different Window sizes

Window Size	P	R	F
2	0.55	0.292	0.381
5	0.54	0.278	0.367

LearningAlgorithm: SkipGram has been chosen as a training algorithm, since it does better for small and medium training data as explained in section 2.8.1.

Iterations (epochs): The default value is between 3 and 50, however, this value has been set to 5 which is already used by other researcher as explained in section 2.8.1.

NegativeSample: this value is set to 25 for optimization purposes in the training task as used by other researchers in same tasks and mentioned in section 2.8.1.

The values for the above parameters are summarized below in Table 4.5.

Table 4.5. Word2Vec Hyper-parameters

Parameter	Value
Layer Size (Dimensions)	50
Window	2
Iterations	5
Negative Samples	25
Learning Algorithm	SkipGram

Word2Vec model will generate a file that contains trained vector for each movie. Each movie will be represented by movie's URI and numerical vector of [1×50 dimension]. The movie's URIs will be preprocessed by lowercasing and removing some symbols. The sample of the generated vectors is shown in Figure 4.8 and the similarity between the movies vectors will be calculated in ranking stage using Cosine distance measure.

httpdbpediaorgresourcegrant_dodwell 0.2828073799610138
0.28735750913619995 **.....** -0.3611583709716797 0.21398897469043732

httpdbpediaorgresourcecaseys_birthday -0.5034074783325195
0.5255025625228882 **.....** -0.6960287094116211 0.2861446738243103

.
.
.

httpdbpediaorgresourcerayful_edmond 0.184496790170669
0.2812623381614685 **.....** -0.480272114276886 0.10692717880010605

httpdbpediaorgresourcedeath_line 0.1056780815124512
0.17926481366157532 -0.09542522579431534 **.....** 0.0865836143493652
-0.3811556100845337

Figure 4.8. Sample of the Generated Trained Vectors

4.5.2 User Representation

The features of the movies found in the user's history will be extracted from DBpedia and LinkedMDB. The features include movie's actors, writers, directors and the genre or the category of the movie. The sample of extracted features of the user's history are illustrated in Figure 4.9

"https://triplydb.com/Triply/linkedmdb/id/actor/31985"
"http://dbpedia.org/resource/Category:American_coming-of-age_films"
"http://dbpedia.org/resource/Category:American_teen_films"
"http://dbpedia.org/resource/Glenn_Close"
"http://dbpedia.org/resource/Category:American_science_fiction_adventure_films"
"https://triplydb.com/Triply/linkedmdb/id/actor/34772"
"http://dbpedia.org/resource/Jack_Nicholson"
"http://dbpedia.org/resource/Category:Films_set_in_the_23rd_century"
"http://dbpedia.org/resource/Category:American_satirical_films"
"https://triplydb.com/Triply/linkedmdb/id/film_genre/31"
"http://dbpedia.org/resource/Sarah_Jessica_Parker"
"https://triplydb.com/Triply/linkedmdb/id/film/38075"

Figure 4.9. Sample of the Extracted Features of the User's History

FOAF vocabulary will be leveraged to build a user model from the features extracted. Two terms of FOAF are used: ***interest*** and ***homepage***. The features: actor URIs, writer URIs, director URIs and the genre URIs stored as object in the RDF statements that represents each user and the predicate is ***foaf:interest*** property as illustrated in Figure 4.10.

<http://RSproject.com/#Person2>
<http://xmlns.com/foaf/0.1/interest>
<https://triplydb.com/Triply/linkedmdb/id/actor/31985>

Figure 4.10. Example of RDF statement for user interest

On the other hand, the URIs of the movies found in the user's history are also stored as object in another RDF statements but the predicate is ***foaf:homepage*** property as shown in Figure 4.11. This property will connect a user with the movies seen previously by him.

```
<http://RSproject.com/#Person2>
<http://xmlns.com/foaf/0.1/homepage>
<http://dbpedia.org/resource/Batman_(1989_film)>.
```

Figure 4.11. Example of RDF statement for movie seen in History

A sample of the user model generated is shown as RDF/XML version in Figure 4.12 and as N-Triples in Figure 4.13.

```
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:j.0="http://xmlns.com/foaf/0.1/">

 <j.0:Person rdf:about="http://RSproject.com/#Person2">
        <j.0:givenname>USER1</j.0:givenname>
       <j.0:homepage
         rdf:resource=" http://dbpedia.org/resource/Mars_Attacks!"/>
         <j.0:homepage
         rdf:resource=" https://triplydb.com/Triply/linkedmdb/id/film/33"/>
         <j.0:homepage
         rdf:resource=" http://dbpedia.org/resource/Batman_(1989_film)"/>
                                          .
                                          .
                                          .
        <j.0:interest
         rdf:resource=" https://triplydb.com/Triply/linkedmdb/id/actor/31985"/>
        <j.0:interest rdf:resource="
           http://dbpedia.org/resource/Category:American_coming-of-
age_films"/>.
         <j.0:interest
         rdf:resource=" http://dbpedia.org/resource/Sarah_Jessica_Parker"/>
 </j.0:Person>
  <j.0:Person rdf:about="http://RSproject.com/#Person3">
                                          .
                                          .
                                          .
</j.0:Person>
                                         ....
</rdf:RDF>
```

Figure 4.12. Sample of the generated FOAF-based User Model (RDF\XML)

```
“<http://RSproject.com/#Person2>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://xmlns.com/foaf/0.1/Person> .”

“<http://RSproject.com/#Person2>
<http://xmlns.com/foaf/0.1/givenname> "USER1" .”

“<http://RSproject.com/#Person2>
<http://xmlns.com/foaf/0.1/homepage>
<http://dbpedia.org/resource/Mars_Attacks!> .”

“<http://RSproject.com/#Person2>
<http://xmlns.com/foaf/0.1/homepage>
<http://dbpedia.org/resource/Maverick_(film)> .”

“<http://RSproject.com/#Person2>
 <http://xmlns.com/foaf/0.1/homepage>
<https://triplydb.com/Triply/linkedmdb/id/film/33> .”

“<http://RSproject.com/#Person2>
 <http://xmlns.com/foaf/0.1/homepage>
<https://triplydb.com/Triply/linkedmdb/id/film/1364> .”

“<http://RSproject.com/#Person2>
<http://xmlns.com/foaf/0.1/homepage>
<http://dbpedia.org/resource/Batman_(1989_film)> .”

“<http://RSproject.com/#Person2>
 <http://xmlns.com/foaf/0.1/interest>
<https://triplydb.com/Triply/linkedmdb/id/actor/31985> .”

“<http://RSproject.com/#Person2>
 <http://xmlns.com/foaf/0.1/interest>
<http://dbpedia.org/resource/Category:American_coming-of-age_films> .”

“<http://RSproject.com/#Person2>
 <http://xmlns.com/foaf/0.1/interest>
<http://dbpedia.org/resource/Sarah_Jessica_Parker> .”
                                        .
                                        .
                                        .
```

Figure 4.13. Sample of the Generated FOAF-based User Model (N-Triples)

4.5.3 Recommendation Process

Now, that the representation for each user's history and item's features are available, so the recommendation process can be started. The recommendation stage contains three steps: generation of candidates, results ranking, and presenting the Top-N items.

A. Generation of Candidates

To generate a list of candidates for a user whose history illustrated previously in Figure 4.6, three SPARQL queries are executed. The first query will extract from LinkedMDB all movies that have same feature in the object of the RDF statement. For example, one of the features in Figure 4.9 is "actor/34772" which is "Michael Ironside actor", therefore a SPARQL query, shown in Figure 4.14, will be executed to extract all the movies that are acted by "Michael Ironside".

```
Select distinct ?s
Where{
        ?s    <https://triplydb.com/Triply/linkedmdb/vocab/actor>
              <https://triplydb.com/Triply/linkedmdb/id/actor/34772>
}
```

Figure 4.14. A SPARQL Query to extract all movies acted by a specific actor

The results of this query is shown below in first column in Table 4.6.

Table 4.6. Extracting Movies that have the actor "Michael Ironside"

?s	Movie's name
"https://triplydb.com/Triply/linkedmdb/id/film/38231"	Total Recall
"https://triplydb.com/Triply/linkedmdb/id/film/39293"	Scanners
"https://triplydb.com/Triply/linkedmdb/id/film/479"	Top Gun

"https://triplydb.com/Triply/linkedmdb/id/film/48104"	Reeker
"https://triplydb.com/Triply/linkedmdb/id/film/58908"	The Veteran
"https://triplydb.com/Triply/linkedmdb/id/film/59105"	Disaster Zone: Volcano in New York

In addition, there are other features such as:

- Category:American_science_fiction_adventure_films
- http://dbpedia.org/resource/Glenn_Close

Another SPARQL query will be executed on DBpedia to extract all the movies that are "American Science fiction adventure movies" and "Gleen Close" as actress in the movie. The third query will generate a list of candidates that have direct link with movies in the history of the user. For example one of the movies in the history of the user is

- "http://dbpedia.org/resource/Batman_(1989_film)"

This movie is also stored as a value of the ***homepage*** property in the FOAF model as shown in Figures 4.12 and 4.13, therefore it can be retrieved and execute the SPARQL query shown below in Figure 4.15 to extract all resources of type Film that are directly linked to ***Batman*** movie.

```
PREFIX dbont: <http://dbpedia.org/ontology/>
Select distinct ?s
Where{
        ?s    ?p    < http://dbpedia.org/resource/Batman_(1989_film)>;
              <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
              dbont:Film
}
```

Figure 4.15. A SPARQL Query to extract all directly linked movies

B. Ranking and Presenting Top-N

The generated candidates need a ranking process in order to recommend the Top-N items. Two similarity measures are used to set a similarity score for each candidate. The first measure is the proposed ELDSD measure which is a link-based measure while the other is a feature-based measure using the cosine of the angle between the vectorized features.

A hybrid approach is proposed that combines both measures to find the similarity between each candidate and the movies seen in history. ELDSD will be used in case that the movies in linked data are strongly linked by direct and indirect links. If the movies are strongly linked by different kinds of links, the distance score will be close to zero. However, if the distance is higher than a certain threshold, it means that the two films are not connected with sufficient relationships, therefore, the measure that depends on the existing of links is useless and the other measure which is feature-based is used. Different values for threshold has been taken from 0.1 to 0.9, but only the results until 0.5 are shown in Table 4.7, because there is no enhancement in terms of P, R, and F-measure when the threshold crosses 0.2 and the best threshold is 0.1. This means when the ELDSD distance becomes more than 0.1, the proposed approach decides to calculate the cosine of the movie's vectors instead.

Table 4.7. Evaluation of the hybrid measure using different threshold values

Threshold	0.1	0.2	0.3	0.4	0.5
P	**0.62**	0.57	0.56	0.56	0.56
R	**0.344**	0.326	0.318	0.318	0.318
F	**0.443**	0.414	0.405	0.405	0.405

After assigning a score for each candidate, a merge sort is applied to produce the Top-N movies that their distance scores close to zero. Depending on the data that are shown in Figures 4.6-4.13 for item and user representation, the recommended Top-10 movies for the user (USER1) is shown below in Figure 4.16.

1. “http://dbpedia.org/resource/The_Green_Mile_(film)”
2. “http://dbpedia.org/resource/The_Fugitive_(1993_film)”
3. “http://dbpedia.org/resource/Malcolm_X_(1992_film)”
4. “http://dbpedia.org/resource/Primal_Fear_(film)”
5. “http://dbpedia.org/resource/There's_Something_About_Mary”
6. “http://dbpedia.org/resource/Jurassic_Park_(film)”
7. “http://dbpedia.org/resource/Pecker_(film)”
8. “http://dbpedia.org/resource/The_War_(1994_film)”
9. “http://dbpedia.org/resource/Forrest_Gump”
10. “http://dbpedia.org/resource/The_Cider_House_Rules_(film)”

Figure 4.16. Top-N movies recommended to USER1

4.5.4 Recommending Items from Other Domains

This dissertation proposed to recommend items from different domains such as: movie, music, and book. In LinkedMDB there are some features illustrated previously in Figure 4.9 that connect a movie to the featured song in the movie in addition to the name of the singer.

- Subject
- Predicate
- Object

“<https://triplydb.com/Triply/linkedmdb/id/film/10221>
<https://triplydb.com/Triply/linkedmdb/vocab/film_featured_song>
<https://triplydb.com/Triply/linkedmdb/id/film_featured_song/30> .”

“<https://triplydb.com/Triply/linkedmdb/id/film_featured_song/30>
<http://www.w3.org/2000/01/rdf-schema#label>
"The James Bond Theme (Film Featured Song)" .”

“<https://triplydb.com/Triply/linkedmdb/id/film_featured_song/30>
<https://triplydb.com/Triply/linkedmdb/vocab/film_featured_song_performed_by>
"John Barry" .”

“<https://triplydb.com/Triply/linkedmdb/id/film/38182>
<https://triplydb.com/Triply/linkedmdb/vocab/film_featured_song>
<https://triplydb.com/Triply/linkedmdb/id/film_featured_song/54> .”

“<https://triplydb.com/Triply/linkedmdb/id/film_featured_song/54>
<http://www.w3.org/2000/01/rdf-schema#label>
"Everybody's Talkin' (Film Featured Song)" .”

“<https://triplydb.com/Triply/linkedmdb/id/film_featured_song/54>
<https://triplydb.com/Triply/linkedmdb/vocab/film_featured_song_performed_by>
"Nilsson" .”

Figure 4.17. Movies Linked with featured songs

Therefore, if the recommended movies are:

- Dr. No (James Bond)
- Midnight Cowboy

The recommended songs will be:

- The James Bond Theme by John Barry for Dr. No movie
- Everybody’s Talkin’ by Nilsson for Midnight movie

On the other hand, the related books are recommended using the extra links that the current approach produced in interlinking stage as shown in Figure 4.3. By using the *relatedBook* property, a related book can be recommended. For example, the recommended movies in Figure 4.16 (Jurassic park, Forrest Gump, and The Cider House Roles movies) are linked with related books in BNB library, therefore the following recommendation of books and the writers will be provided.

- Jurassic Park Book by Michael Crichton
- Forrest Gump Book by Winston Groom
- The Cider House Rules Book by John Irving

4.6 Evaluation Datasets Description

Two datasets have been used for evaluation stage, the first one is MovieLens 1M and the second is Yahoo! Movie Dataset. The movies in both of them are rated in the rage [1-5] and the movies are linked to the corresponding DBpedia URI.

4.6.1 MovieLens 1M Dataset

MovieLens contains three files: rating.tsv, a Tab Separated Values (TSV) that includes 4 columns: user_ID, movie_ID, rating (1-5), and timestamp (Time of the rating). See Figure 4.18 that shows a sample lines of this file.

" UserID"	"MovieID"	"Rating"	"Timestamp"
1	3186	4	978300019
1	1270	5	978300055
1	1022	5	978300055
1	1721	4	978300055
1	2340	3	978300103
		.	
		.	
		.	
6040	2917	4	997454429
6040	1921	4	997454464
6040	1784	3	997454464
6040	161	3	997454486
6040	1221	4	998315055

Figure 4.18. Sample lines of rating.tsv file

The ***movie_ID*** is described in another file called ***movie.dat*** that includes data about the movies in three columns: ***movie_ID, movie_Title***, and ***Genres***(pipe-separated) as shown in Figure 4.19.

"Movie_ID"::"Movie_Title"::"Genres"
"1::Toy Story (1995)::Animation|Children's|Comedy"
"2::Jumanji (1995)::Adventure|Children's|Fantasy"
"3::Grumpier Old Men (1995)::Comedy|Romance"
"4::Waiting to Exhale (1995)::Comedy|Drama"
"5::Father of the Bride Part II (1995)::Comedy"
.
.
.
"3948::Meet the Parents (2000)::Comedy"
"3949::Requiem for a Dream (2000)::Drama"
"3950::Tigerland (2000)::Drama"
"3951::Two Family House (2000)::Drama"
"3952::Contender, The (2000)::Drama|Thriller"

Figure 4.19. Sample of movie.dat file for movies description

To evaluate a linked data-based RS using MovieLens, there is a file that connects every movie ID in MovieLens to its corresponding DBpedia URI. This file, which its sample is provided in Figure 4.20, includes three columns: ***movie_ID, movie_Title, and DBpedia_URI***. Therefore, it becomes possible to

evaluate the proposed system that recommends DBpedia URIs in the final result using the MovieLens Dataset through this mapping file.

Movie_ID	Movie_Title	DBpedia_URI
781	Stealing Beauty (1996)	"http://dbpedia.org/resource/Stealing_Beauty"
1799	Suicide Kings (1997)	"http://dbpedia.org/resource/Suicide_Kings"
521	Romeo Is Bleeding (1993)	"http://dbpedia.org/resource/Romeo_Is_Bleeding"
3596	Screwed (2000)	"http://dbpedia.org/resource/Screwed_(2000_film)"
3682	Magnum Force (1973)	"http://dbpedia.org/resource/Magnum_Force"
		.
		.
		.
2634	Mummy, The (1959)	"http://dbpedia.org/resource/The_Mummy_(1959_film)"
2242	Grandview, U.S.A. (1984)	"http://dbpedia.org/resource/Grandview,_U.S.A."
2762	Sixth Sense, The (1999)	"http://dbpedia.org/resource/The_Sixth_Sense"

Figure 4.20. Sample Lines of Mapping MovieLens to DBpedia

4.6.2 Yahoo! Movies Dataset

In this dataset, there is no timestamp. Therefore, the rating file has three columns only: user_ID, movie_ID, and rating in [1-5] range as shown in Figure 4.21 below.

" UserID"	"MovieID"	"Rating"
1	1800030906	5.0
1	1800029049	5.0
1	1800373145	4.0
1	1800256362	4.0
	.	
	.	
	.	
7638	1800018796	5.0
7638	1803470438	5.0
7638	1800064675	5.0
7638	1800131296	5.0

Figure 4.21. Sample of the Rating file in Yahoo! movie Dataset

The movie_ID is linked to the DBpedia URI in another file that do the mapping needed in order to evaluate Linked Data-based system as shown below in Figure 4.22.

```
1800026906   "http://dbpedia.org/resource/Following"
1800096627   "http://dbpedia.org/resource/Monkey_Shines"
1800080499   "http://dbpedia.org/resource/In_Search_of_the_Castaways_(film)"
1800136520   "http://dbpedia.org/resource/The_Women_(1939_film)"
                                    .
                                    .
                                    .

1808626957   "http://dbpedia.org/resource/I_Am_Cuba"
1802837000   "http://dbpedia.org/resource/The_Replacements_(film)"
1800149589   "http://dbpedia.org/resource/Woman_in_the_Dunes"
1800026004   "http://dbpedia.org/resource/The_Proprietor"
```

Figure 4.22. Sample Lines of Mapping Yahoo! movie Dataset to DBpedia

4.7 Evaluation Setting and Protocol

Offline evaluation is adapted using the two mentioned datasets after splitting them into two sets: training set 70% and testing set 30%. For testing, "All unrated items protocol" and "rated test items protocol" are used. Local Rank-based precision, recall, f-measure, and MRR are calculated for each user. After that, the results are averaged over all users.

The training data for each user is used to build a user model in order to generate the Top-N relevant movies for every user. The generated Top-N items will then be compared to the test data for that user in order to calculate precision, which is the ability of RS to suggest movies that are truly relevant to users; recall, which measures the RS's ability to gather relevant movies for that specific user; F-measure, which is a harmonic mean between precision and recall; and MRR, which is how early a relevant result appears among ranked results. The relevant items for each user are determined by finding the mean of all rated items for each user and

consider all movies that are rated with a rating value greater than the calculated mean as relevant items for that user.

4.8 Experimental Results

Two types of experiments are conducted in this dissertation as follows:

1. Evaluation of items generated from LOD that are not limited to the items available in MovieLens and Yahoo! movie datasets.This experiment is called "All unrated items protocol" and it is expected to have low metrics values, since the movies that were suggested may not appear at all for the user to express his opinion and give his rating about them. However, the comparisons will be with other approaches under the same environment and evaluation setting for getting a relative comparison of approaches.
2. Evaluation of recommended items that are limited to the items in MovieLens and Yahoo datasets which is "rated test items protocol".

4.8.1 Experiments 1

In the first experiment, the recommended items obtained from LOD datasets by applying the different kinds of the proposed algorithms that utilized direct and indirect relations. This experiment is expected to have low precision and recall because some of the generated items are not known in MovieLens and Yahoo! Movie datasets. However, we make relative comparisons between the different approaches and the comparisons are done under the same evaluation setting and same circumstances.

Firstly, we conduct the evaluation process on MovieLens and the generated items are not limited to the testing set. The dataset is sorted according to the time attribute in order to simulate the behavior of the RS. Several experiments are conducted to evaluate the proposed approach and compared with other approaches

that use different pure link based similarity measures such as LDSD and TLDSD and approaches that depend on common and characteristic features between the items such as PICSS and EPICS for ranking task. To make fair comparisons, LDS java library (explained in Section 2.9) is used. Precision and Recall for Top-5 and Top-10 recommendations are calculated and presented in Tables 4.8 - 4.10.

Table 4.8. Precision measure (MovieLens)

	LDSD[8]	TLDSD[9]	PICSS[11]	EPICS[12]	Proposed approach
Precision@5	0.04	0.0329	0.031	0.02	**0.06**
Precision@10	0.03	0.0322	0.031	0.027	**0.05**

Table 4.9. Recall measure (MovieLens)

	LDSD[8]	TLDSD[9]	PICSS[11]	EPICS[12]	Proposed approach
Recall@5	0.062	0.05	0.05	0.025	**0.08**
Recall@10	0.1	0.087	0.087	0.075	**0.12**

Table 4.10. F-measure (MovieLens)

	LDSD [8]	TLDSD[9]	PICSS[11]	EPICS[12]	Proposed approach
f-measure@5	0.0487	0.0396	0.0384	0.0227	**0.0731**
f-measure@10	0.0503	0.0471	0.0463	0.0407	**0.0735**

As noticed, the proposed approach outperforms other approaches in terms of precision, recall and f-measure with enhancement between 2-3% for f-measure and the comparison is clarified as a bar chart in Figure 4.23.

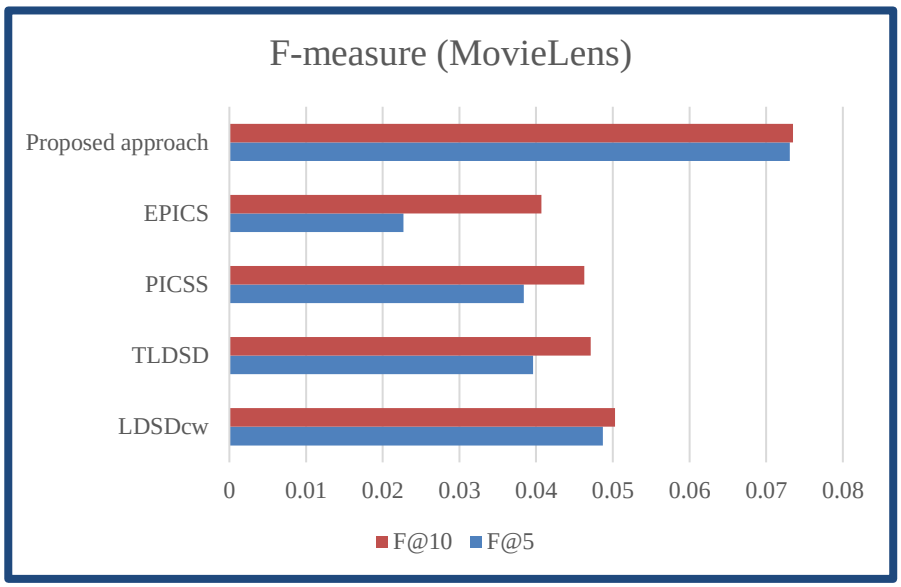

Figure 4.23. F-measure comparisons (MovieLens)

Secondly, the evaluation process is conducted on Yahoo! Movie dataset and the generated items obtained from LOD and also they are not limited to the testing set in Yahoo! dataset. Different evaluation metrics are conducted as shown in Tables 4.11- 4.13.

Table 4.11. Precision (Yahoo! Movie Dataset)

	LDSD[8]	TLDSD[9]	PICSS[11]	EPICS[12]	Proposed approach
Precision@5	0.04	0.05	0.04	0.04	0.05
Precision@10	0.06	0.06	0.03	0.03	**0.07**

Table 4.12. Recall (Yahoo! Movie Dataset)

	LDSD[8]	TLDSD[9]	PICSS[11]	EPICS[12]	Proposed approach
Recall@5	0.09	0.07	0.08	0.07	**0.10**
Recall@10	0.19	0.16	0.1	0.09	**0.19**

Table 4.13. F-measure (Yahoo! Movie dataset)

	LDSD[8]	TLDSD[9]	PICSS[11]	EPICS[12]	Proposed approach
f-measure@5	0.059	0.062	0.062	0.054	**0.067**
f-measure@10	0.098	0.091	0.055	0.045	**0.1**

The results show that experiment on Yahoo! dataset also outperforms the other approaches in terms of P, R, and f-measure. Figure 4.24 bellow shows the enhancement as a bar chart.

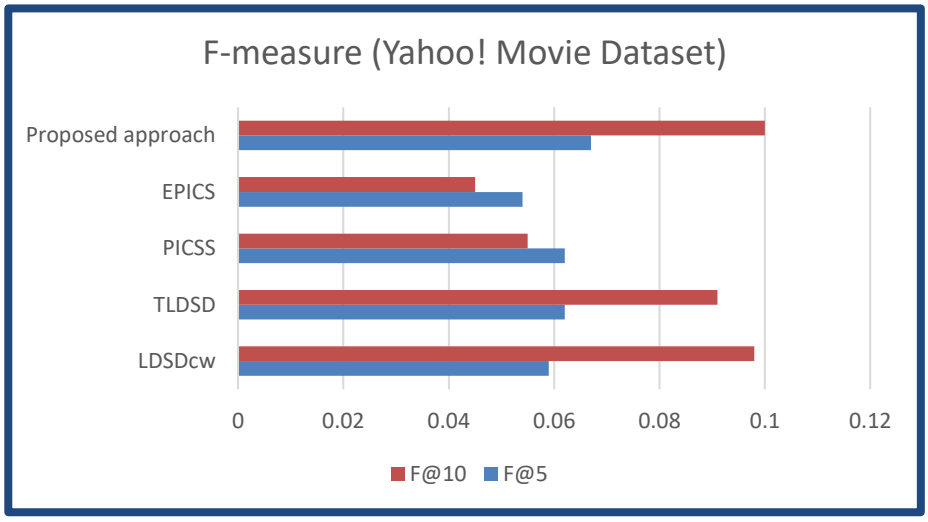

Figure 4.24. F-measure comparison (Yahoo! Movie Dataset)

Precision and recall do not consider the ordering of the recommendation items. Therefore, Mean Reciprocal Rank (MRR) is calculated and the results are shown below in Table 4.14 and the differences are illustrated in Figure 4.25.

Table 4.14. MRR for Both datasets

	LDSD[8]	TLDSD[9]	PICSS[11]	EPICS[12]	Proposed approach
MRR MovieLens	0.094	0.062	0.056	0.047	**0.105**
MRR Yahoo	0.10	0.077	0.059	0.052	**0.10**

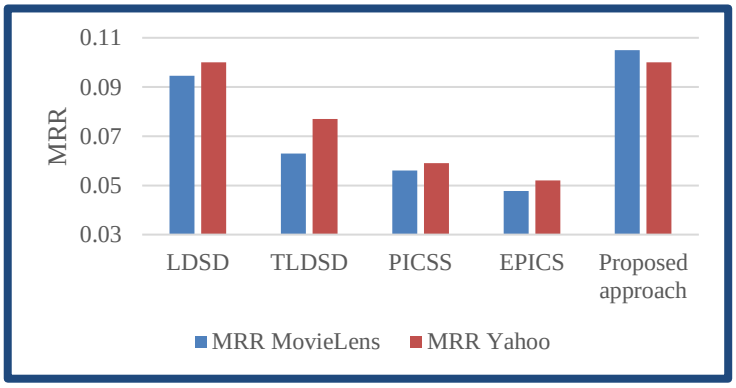

Figure 4.25. MRR for different approaches (Both datasets)

4.8.2 Experiment 2

In the second experiment, first, we conduct the evaluation process on MovieLens and the generated items are limited to the testing set. The proposed approach is evaluated and compared with the same approaches mentioned in first experiment which are LDSD,TLDSD, PICSS, and EPICS and using the same LDS Java library and same evaluation environment for all methods. Precision and Recall for Top-10 recommendations are also calculated on different K cutoff and presented in Table 4.15 and 4.16, respectively. Gradient colors are applied to the table to highlight the highest value with green and lowest value with red.

Table 4.15. Precision at different N cutoff (MovieLens)

Precision	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10
LDSD[8]	0.7	0.65	0.66	0.7	0.68	0.66	0.62	0.58	0.55	0.56
TLDSD[9]	0.5	0.7	0.66	0.6	0.58	0.51	0.54	0.55	0.53	0.53
PICSS[11]	0.6	0.6	0.5	0.47	0.47	0.48	0.48	0.47	0.46	0.47
EPICS[12]	0.4	0.55	0.56	0.62	0.6	0.58	0.55	0.52	0.53	0.52
Proposed Approach	0.7	0.75	0.7	0.7	0.7	0.65	0.64	0.6	0.56	0.57

Highest Precision — Lowest Precision

Table 4.16. Recall at different N cutoff (MovieLens)

Recall	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10
LDSD[8]	0.03	0.06	0.11	0.15	0.19	0.22	0.24	0.26	0.28	0.31
TLDSD[9]	0.02	0.08	0.1	0.13	0.16	0.17	0.21	0.24	0.26	0.29
PICSS[11]	0.02	0.06	0.08	0.11	0.14	0.17	0.2	0.22	0.24	0.27
EPICS[12]	0.02	0.06	0.1	0.14	0.16	0.19	0.21	0.22	0.25	0.27
Proposed Approach	0.03	0.075	0.11	0.15	0.19	0.22	0.26	0.27	0.294	0.32

Highest Recall — Lowest Recall

Figure 4.26 below illustrates the F-measure curves for different approaches on different K cutoff.

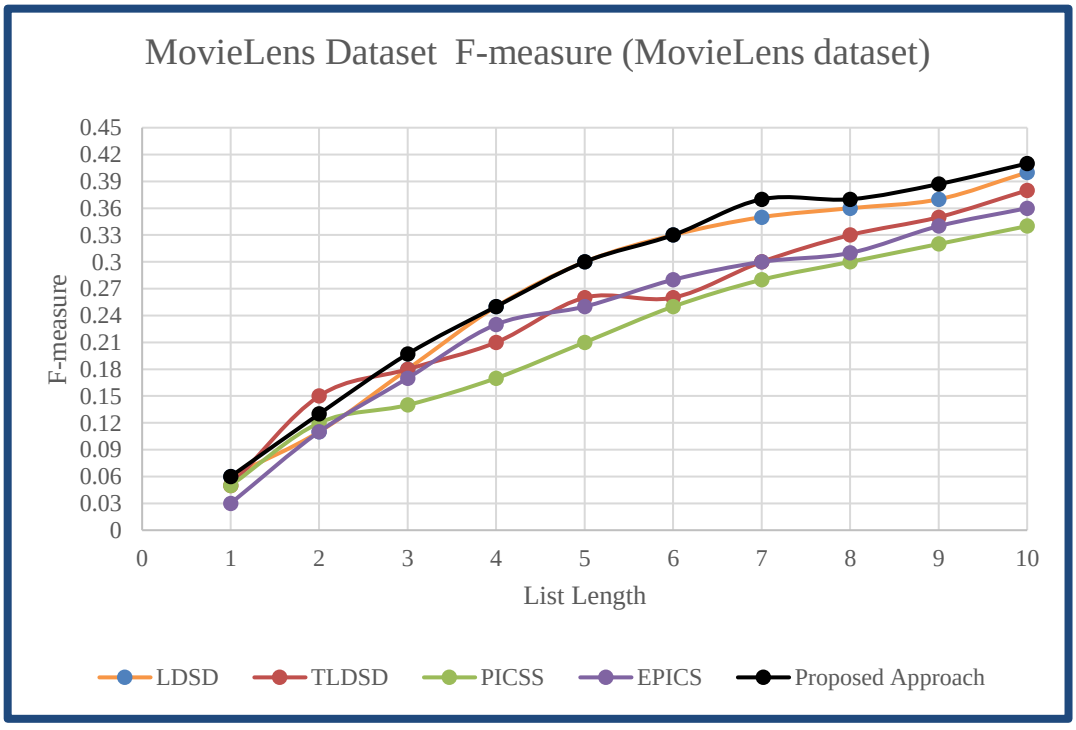

Figure 4.26. F-measure comparisons for different approaches

The proposed hybrid approach that combines ELDSD and cosine distance of the vectorized features outperforms the link-based measures with up to with up to 7.8% improvement in f-measure and with up to 20.5% improvement over other approaches that depend on the features only.

Second, the evaluation process is conducted on Yahoo! Movie dataset and the generated items are limited to the testing set. Precision and Recall for Top-10 recommendations are also calculated on different K cutoff and presented in Table 4.17 and 4.18, respectively. Every cell is also colored to highlight the highest values.

Table 4.17. Precision in different N cutoffs (Yahoo! Dataset)

	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10
LDSD[8]	0.7	0.6	0.63	0.65	0.64	0.68	0.642	0.587	0.6	0.6
TLDSD[9]	0.7	0.65	0.69	0.65	0.65	0.65	0.642	0.65	0.6	0.59
PICSS[11]	0.7	0.7	0.6	0.52	0.52	0.48	0.5	0.5	0.52	0.52
EPICS[12]	0.6	0.6	0.56	0.57	0.54	0.48	0.47	0.52	0.51	0.5
Proposed Approach	0.7	0.7	0.73	0.7	0.66	0.66	0.671	0.65	0.64	0.63

Highest Precision — Lowest Precision

Table 4.18. Recall for different N cutoffs (Yahoo Dataset)

	@1	@2	@3	@4	@5	@6	@7	@8	@9	@10
LDSD[8]	0.04	0.07	0.119	0.156	0.193	0.249	0.278	0.291	0.34	0.38
TLDSD[9]	0.04	0.08	0.13	0.16	0.2	0.24	0.28	0.32	0.35	0.37
PICSS[11]	0.04	0.08	0.11	0.13	0.16	0.18	0.21	0.25	0.28	0.31
EPICS[12]	0.04	0.07	0.1	0.14	0.16	0.17	0.2	0.25	0.28	0.3
Proposed Approach	0.04	0.08	0.135	0.173	0.203	0.248	0.289	0.324	0.366	0.399

Highest Recall — Lowest Recall

The harmonic mean of precision and recall, which is called f-measure is also calculated on different K cutoffs as shown in Figure 4.27.

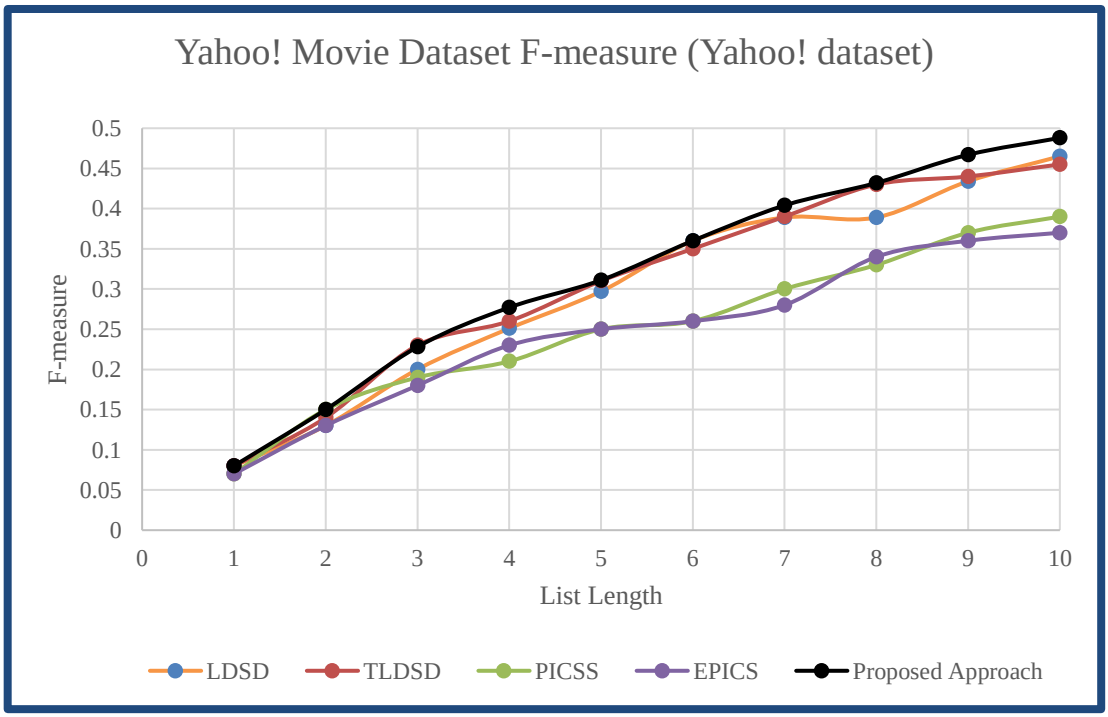

Figure 4.27. F-measure comparisons for different approaches

By using Yahoo Movie dataset for evaluation purposes, the proposed hybrid approach is also outperforms the link-based measures with up to with up to 7.2% improvement in f-measure and with up to 31.8% improvement over other approaches that depend on the features only.

As in the first experiment, Mean Reciprocal Rank (MRR) is calculated to evaluate the order of the relevant items obtained and the results for both datasets are shown below in Table 4.19 and the differences are illustrated in Figure 4.28.

Table 4.19. MRR for both datasets

	LDSD[8]	TLDSD[9]	PICSS[11]	EPICS[12]	Proposed approach
MRR MovieLens	0.786	0.75	0.783	0.658	**0.833**
MRR Yahoo	0.783	0.8	0.8	0.733	**0.8**

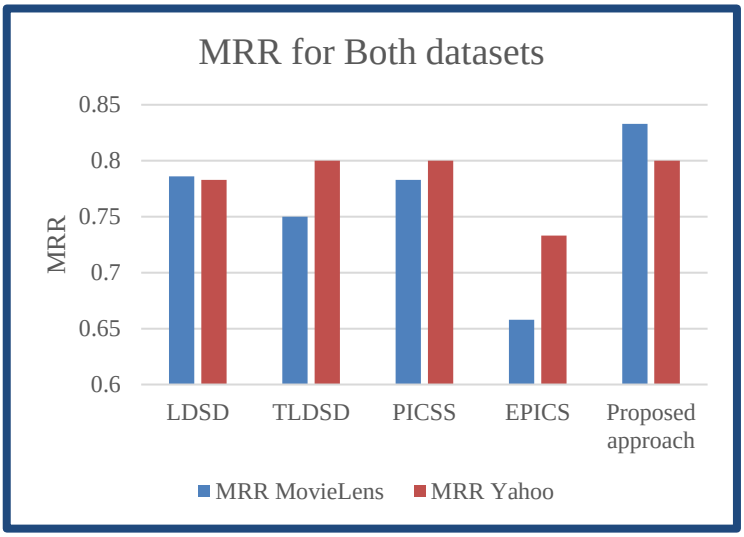

Figure 4.28. MRR for different approaches

Precision-recall curve is illustrated below in Figure 4.29 for the different approaches on different N cutoff using MovieLens dataset and the area under the curve is shown in Table 4.20.

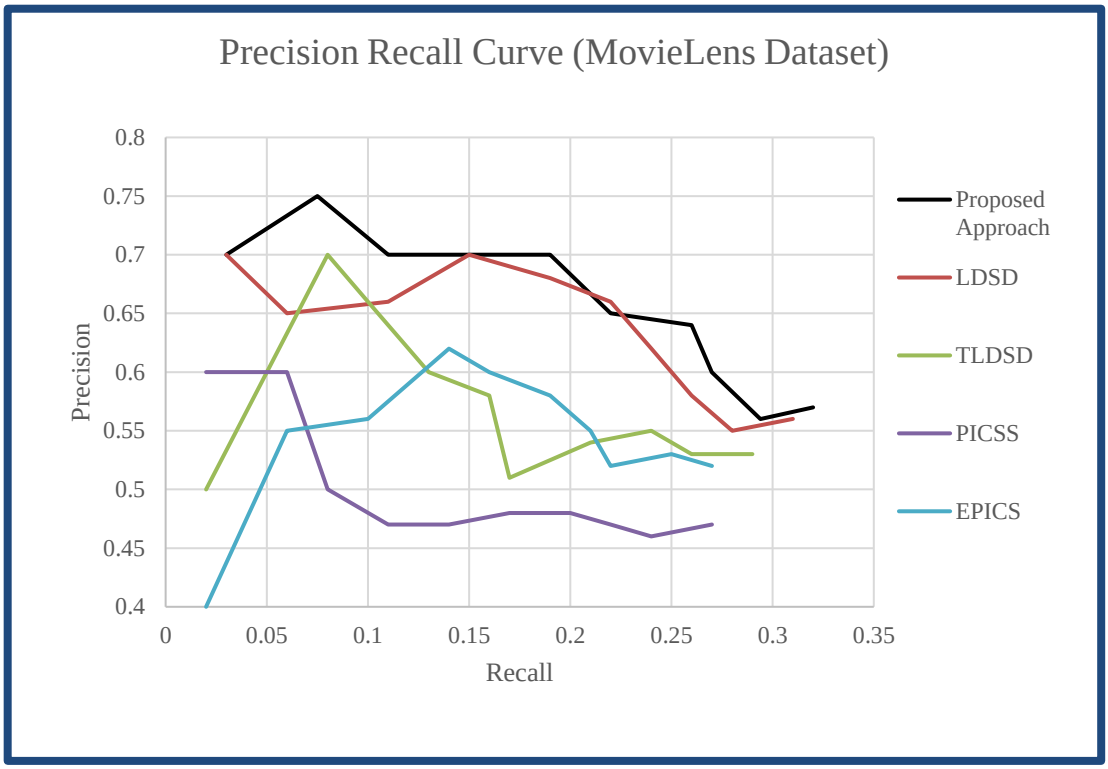

Figure 4.29. Precision Recall Curve for different approaches (MovieLens)

Table 4.20. Area under the P R curve for different approaches (MovieLens)

	AUPRC
LDSD[8]	0.18
TLDSD[9]	0.16
PICSS[11]	0.13
EPICS[12]	0.14
Proposed Approach	**0.19**

The above Table 4.20 shows that the proposed approach outperforms other approaches with area under the curve reaches 0.19 for the proposed approach.

Another similar experiment is done using Yahoo! Movie dataset and the curves are shown in Figure 4.30.

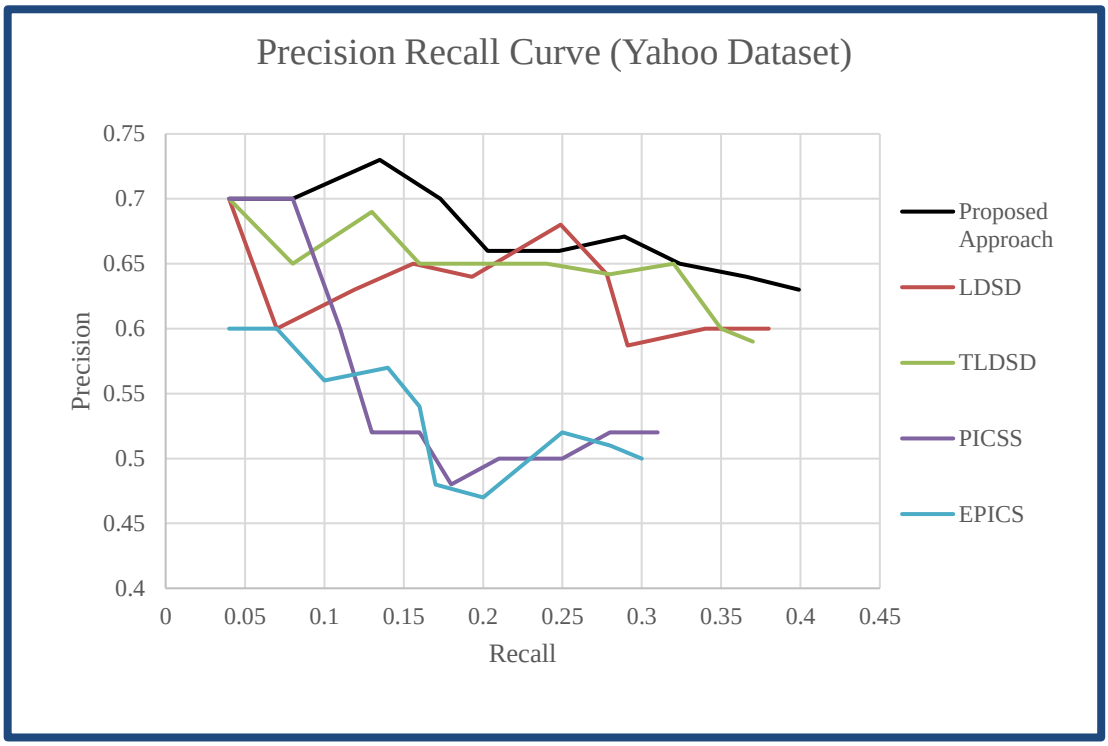

Figure 4.30. Precision Recall Curve for different approaches (Yahoo)

In addition, the area under the curve shown in Table 4.21 presents a clear excellence for the proposed approach with highest value reaches 0.24 in comparison with only 0.21 for LDSD and TLDSD and 0.15 for PICSS while EPICS achieves the lowest value with 0.139 only.

Table 4.21. Area under the P R curve for different approaches (Yahoo)

	AUPRC
LDSD[8]	0.214
TLDSD[9]	0.215
PICSS[11]	0.15
EPICS[12]	0.139
Proposed Approach	**0.24**

CHAPTER FIVE

CONCLUSION AND FUTURE WORKS

5.1 Conclusion

In this dissertation, a brief review of the current state of Linked Open Data in Recommendation System is presented and the benefit of using LOD and the methods that are used by previous works to exploit LOD for recommendation tasks are identified. In addition, the different RSs techniques are highlighted and the main challenges in developing such systems are clarified. The conclusions drawn by the thesis can be summarized as follows

1- The use of LOD has a good impact on enhancing the results of RS in different domains due to the huge knowledge exist and many studies can be conducted to exploit the different kinds of relations that are evolved overtime and provide us with new knowledge to utilize for recommendation tasks and any other task that need external knowledge.

2- An extended version of LinkedMDB dataset can help other researchers in different kinds of applications who need to utilize rich features of the movies. In addition to the ability of obtaining all the features of the related books.

3- ELDSD enhances the traditional LDSD measure and can be used by other researchers to find the similarity of the resources in any domain.

4- It is noted that the pure link-based measures such as (LDSD and TLDSD) that depend on the relatedness of the resources gained better results than the feature-based measures such as (PICSS and EPICS) that depend on the similarity of the resources' features and this reflects the significance of the structure of the existing links.

5- Moreover, the proposed hybrid measure that depends on similarity and relatedness outperforms the other approaches. Other researchers can use this measure, especially, in some domains that have weakly linked resources or few features exist.

5.2 Limitations

- The performance of the proposed system needs to be improved because of the computational complexity in analyzing the items and their relations exist in the datasets. In addition, the remote endpoint of the DBpedia limits number of results and has a poor performance to access them.
- Some movies in MovieLens and Yahoo Movie datasets are not mapped to their corresponding DBpedia URI and this affects the effectiveness of this work.
- Although we find the best value for K in ELDSD measure for both datasets used, there is difficulties in finding k. The choosing of K value depends on the used dataset and domain.
- The proposed system could not be able to measure the quality of the vectorized features and extrinsic evaluation is done, which evaluate the generated embedding through a specific task which is a recommendation task.
- The absence of publicly available datasets including user evaluations of various kinds of items from different domains for use in evaluating a multi-domain RS.

5.3 Future Works

- An automation of selecting the value of K in ELDSD equation is needed and we leave it as a future work.
- The mapping file that maps MovieLens movies to their corresponding DBpedia URI need to be revised to fix some mistakes in the matched movies and need to be updated to fit with other versions of MovieLens such as MovieLens 10M and MovieLens 20M.

- Evaluation dataset is needed to directly evaluate the generated embeddings
- Various experiments will be conducted with other evaluation metrics such as Diversity and Novelty.
- Other datasets from other domains such as books and musics will be used as a future work.

References

REFERENCES

[1] P. Groth, F. van Harmelen, R. Hoekstra, and G. Antoniou, "A semantic web primer," ed: MIT Press, 2012.

[2] C. Figueroa, Corrales, J. C., & Morisio, M. , *RECOMMENDER SYSTEMS BASED ON LINKED DATA(1st ed.)*. Universidad del Cauca., 2019.

[3] W. Ali, M. Saleem, B. Yao, A. Hogan, and A.-C. N. Ngomo, "A survey of RDF stores & SPARQL engines for querying knowledge graphs," *The VLDB Journal,* pp. 1-26, 2021.

[4] T. Di Noia and V. C. Ostuni, "Recommender systems and linked open data," in *Reasoning Web International Summer School*, 2015: Springer, pp. 88-113.

[5] E. Rajabi and W. Greller, "Exposing Social Data as Linked Data in Education," *International Journal on Semantic Web and Information Systems (IJSWIS)*, vol. 15, no. 2, pp. 92-106, 2019.

[6] X. Cai, Z. Hu, P. Zhao, W. Zhang, and J. Chen, "A hybrid recommendation system with many-objective evolutionary algorithm," *Expert Systems with Applications,* vol. 159, p. 113648, 2020.

[7] F. De la Prieta *et al.*, *Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection*. Springer, 2016.

[8] A. Passant, "Measuring Semantic Distance on Linking Data and Using it for Resources Recommendations," in *AAAI spring symposium: linked data meets artificial intelligence*, 2010, vol. 77, p. 123.

[9] S. Alfarhood, K. Labille, and S. Gauch, "PLDSD: propagated linked data semantic distance," in *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, 2017: IEEE, pp. 278-283.

[10] S. Alfarhood, S. Gauch, and K. Labille, "Semantic Distance Spreading Across Entities in Linked Open Data," *Information,* vol. 10, no. 1, p. 15, 2019.

[11] R. Meymandpour and J. G. Davis, "A semantic similarity measure for linked data: An information content-based approach," *Knowledge-Based Systems,* vol. 109, pp. 276-293, 2016.

[12] F. Komeiha, N. Cheniki, Y. Sam, A. Jaber, N. Messai, and T. Devogele, "LDS: Java Library for Linked Open Data Based Similarity Measures," in *2020 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2020: IEEE, pp. 476-481.

[13] M. Boubenia, A. Belkhir, and F. M. h. Bouyakoub, "Combining Linked Open Data Similarity and Relatedness for Cross OSN Recommendation," *International Journal on Semantic Web and Information Systems (IJSWIS),* vol. 16, no. 2, pp. 59-90, 2020.

[14] G. O. M. D. Silva, P. R. D. Souza, and F. A. Durão, "HSLD: a hybrid similarity measure for linked data resources," *International Journal of Metadata, Semantics and Ontologies,* vol. 14, no. 1, pp. 16-25, 2020.

[15] U. Yadav, N. Duhan, and K. K. Bhatia, "Dealing with Pure New User Cold-Start Problem in Recommendation System Based on Linked Open Data and Social Network Features," *Mobile Information Systems,* vol. 2020, 2020.

[16] H. Sebbaq, N.-e. el Faddouli, and S. Bennani, "Recommender System to Support MOOCs Teachers: Framework based on Ontology and Linked Data," in *Proceedings of*

the 13th International Conference on Intelligent Systems: Theories and Applications, 2020, pp. 1-7.

[17] C. Musto, P. Lops, M. de Gemmis, and G. Semeraro, "Semantics-aware Recommender Systems exploiting Linked Open Data and graph-based features," *Knowledge-Based Systems*, vol. 136, pp. 1-14, 2017.

[18] A. Iana, S. Jung, P. Naeser, A. Birukou, S. Hertling, and H. Paulheim, "Building a conference recommender system based on SciGraph and WikiCFP," in *International Conference on Semantic Systems*, 2019: Springer, pp. 117-123.

[19] S. Natarajan, S. Vairavasundaram, S. Natarajan, and A. H. Gandomi, "Resolving data sparsity and cold start problem in collaborative filtering recommender system using Linked Open Data," *Expert Systems with Applications*, vol. 149, p. 113248, 2020.

[20] J. Oliveira, C. Delgado, and A. C. Assaife, "A recommendation approach for consuming linked open data," *Expert Systems with Applications*, vol. 72, pp. 407-420, 2017.

[21] I. Vagliano, "Content Recommendation Through Linked Data," PhD, Computer and Control Engineering, Politecnico di Torino, 2017.

[22] T. Di Noia, C. Magarelli, A. Maurino, M. Palmonari, and A. Rula, "Using ontology-based data summarization to develop semantics-aware recommender systems," in *European Semantic Web Conference*, 2018: Springer, pp. 128-144.

[23] N. S. Selvan, S. Vairavasundaram, and L. Ravi, "Fuzzy ontology-based personalized recommendation for internet of medical things with linked open data," *Journal of Intelligent & Fuzzy Systems*, no. Preprint, pp. 1-11, 2019.

[24] I. C. Hsu and Y. H. Lin, "Integrated machine learning with semantic web for open government data recommendation based on cloud computing," *Software: Practice and Experience*, vol. 50, no. 12, pp. 2293-2312, 2020.

[25] J. Li and Z. Ye, "Course recommendations in online education based on collaborative filtering recommendation algorithm," *Complexity*, vol. 2020, 2020.

[26] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender systems handbook*: Springer, 2011, pp. 1-35.

[27] F. Tahmasebi, M. Meghdadi, S. Ahmadian, and K. Valiallahi, "A hybrid recommendation system based on profile expansion technique to alleviate cold start problem," *Multimedia Tools and Applications*, vol. 80, no. 2, pp. 2339-2354, 2021.

[28] P. Yochum, L. Chang, T. Gu, and M. Zhu, "Linked open data in location-based recommendation system on tourism domain: A survey," *IEEE Access*, vol. 8, pp. 16409-16439, 2020.

[29] R. Wang, H. K. Cheng, Y. Jiang, and J. Lou, "A novel matrix factorization model for recommendation with LOD-based semantic similarity measure," *Expert Systems with Applications*, vol. 123, pp. 70-81, 2019.

[30] H. Zhang, I. Ganchev, N. S. Nikolov, and M. Stevenson, "UserReg: A simple but strong model for rating prediction," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021: IEEE, pp. 3595-3599.

[31] G. Guo, J. Zhang, and N. Yorke-Smith, "A novel recommendation model regularized with user trust and item ratings," *ieee transactions on knowledge and data engineering*, vol. 28, no. 7, pp. 1607-1620, 2016.

[32] J. Bennett and S. Lanning, "The netflix prize," in *Proceedings of KDD cup and workshop*, 2007, vol. 2007: New York, NY, USA., p. 35.

[33] Z. Cheng, X. Chang, L. Zhu, R. C. Kanjirathinkal, and M. Kankanhalli, "MMALFM: Explainable recommendation by leveraging reviews and images," *ACM Transactions on Information Systems (TOIS)*, vol. 37, no. 2, pp. 1-28, 2019.
[34] V. W. Anelli, "Knowledge-Enabled Recommender Systems in the Linked Data Era," 2019.
[35] E. Christakopoulou and G. Karypis, "Local item-item models for top-n recommendation," in *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016, pp. 67-74.
[36] S. Reddy, S. Nalluri, S. Kunisetti, S. Ashok, and B. Venkatesh, "Content-based movie recommendation system using genre correlation," in *Smart Intelligent Computing and Applications*: Springer, 2019, pp. 391-397.
[37] Z. Fayyaz, M. Ebrahimian, D. Nawara, A. Ibrahim, and R. Kashef, "Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities," *applied sciences*, vol. 10, no. 21, p. 7748, 2020.
[38] U. Javed, K. Shaukat, I. A. Hameed, F. Iqbal, T. M. Alam, and S. Luo, "A review of content-based and context-based recommendation systems," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 16, no. 3, pp. 274-306, 2021.
[39] M. K. Najafabadi, A. Mohamed, and C. W. Onn, "An impact of time and item influencer in collaborative filtering recommendations using graph-based model," *Information Processing & Management*, vol. 56, no. 3, pp. 526-540, 2019.
[40] H. Fang, C. Chen, Y. Long, G. Xu, and Y. Xiao, "DTCRSKG: A Deep Travel Conversational Recommender System Incorporating Knowledge Graph," *Mathematics*, vol. 10, no. 9, p. 1402, 2022.
[41] S. Souabi, A. Retbi, M. K. I. K. Idrissi, and S. Bennani, "Recommendation Systems on E-Learning and Social Learning: A Systematic Review," *Electronic Journal of e-Learning*, vol. 19, no. 5, pp. pp432-451, 2021.
[42] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, "A content-based recommender system for computer science publications," *Knowledge-Based Systems*, vol. 157, pp. 1-9, 2018.
[43] M. del CarmenRodríguez-Hernández, R. del-Hoyo-Alonso, S. Ilarri, R. M. Montafñés-Salas, and S. Sabroso-Lasa, "An Experimental Evaluation of Content-based Recommendation Systems: Can Linked Data and BERT Help?," in *2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, 2020: IEEE, pp. 1-8.
[44] Q. Zhang, G. Zhang, J. Lu, and D. Wu, "A framework of hybrid recommender system for personalized clinical prescription," in *2015 10th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, 2015: IEEE, pp. 189-195.
[45] P. Lops, D. Jannach, C. Musto, T. Bogers, and M. Koolen, "Trends in content-based recommendation," *User Modeling and User-Adapted Interaction*, vol. 29, no. 2, pp. 239-249, 2019.
[46] C. Figueroa *et al.*, "Executing, Comparing, and Reusing Linked-Data-Based Recommendation Algorithms with the Allied Framework," in *Semantic Web Science and Real-World Applications*: IGI Global, 2019, pp. 18-47.
[47] J. K. Tarus, Z. Niu, and G. Mustafa, "Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning," *Artificial intelligence review*, vol. 50, no. 1, pp. 21-48, 2018.

[48] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[49] R. L. Rosa, G. M. Schwartz, W. V. Ruggiero, and D. Z. Rodríguez, "A knowledge-based recommendation system that includes sentiment analysis and deep learning," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2124-2135, 2018.

[50] L. Peska and P. Vojtas, "Enhancing recommender system with linked open data," in *International Conference on Flexible Query Answering Systems*, 2013: Springer, pp. 483-494.

[51] T. Di Noia, "Knowledge-enabled Recommender Systems: Models, Challenges, Solutions," in *KDWeb*, 2017.

[52] A. Carbonaro, "Concept Integration to Develop Next Generation of Technology-Enhanced Learning Systems," in *Project and Design Literacy as Cornerstones of Smart Education*: Springer, 2020, pp. 121-129.

[53] M.-Y. Hsieh, W.-K. Chou, and K.-C. Li, "Building a mobile movie recommendation service by user rating and APP usage with linked data on Hadoop," *Multimedia Tools and Applications*, vol. 76, no. 3, pp. 3383-3401, 2017.

[54] G. Geetha, M. Safa, C. Fancy, and D. Saranya, "A hybrid approach using collaborative filtering and content based filtering for recommender system," in *Journal of Physics: Conference Series*, 2018, vol. 1000, no. 1: IOP Publishing, p. 012101.

[55] R. Meymandpour and J. G. Davis, "Enhancing recommender systems using linked open data-based semantic analysis of items," in *Proceedings of the 3rd australasian web conference (AWC 2015)*, 2015, vol. 27, pp. 11-17.

[56] N. AlRossais, D. Kudenko, and T. Yuan, "Improving cold-start recommendations using item-based stereotypes," *User Modeling and User-Adapted Interaction*, vol. 31, no. 5, pp. 867-905, 2021.

[57] P. Kumar and R. S. Thakur, "Recommendation system techniques and related issues: a survey," *International Journal of Information Technology*, vol. 10, no. 4, pp. 495-501, 2018.

[58] C. Yan, Y. Chen, and L. Zhou, "Differentiated fashion recommendation using knowledge graph and data augmentation," *Ieee Access*, vol. 7, pp. 102239-102248, 2019.

[59] E. Çano and M. Morisio, "Hybrid recommender systems: A systematic literature review," *Intelligent Data Analysis*, vol. 21, no. 6, pp. 1487-1524, 2017.

[60] J. Shokeen and C. Rana, "A study on features of social recommender systems," *Artificial Intelligence Review*, vol. 53, no. 2, pp. 965-988, 2020.

[61] B. Heinrich, M. Hopf, D. Lohninger, A. Schiller, and M. Szubartowicz, "Data quality in recommender systems: the impact of completeness of item content data on prediction accuracy of recommender systems," *Electronic Markets*, vol. 31, no. 2, pp. 389-409, 2021.

[62] Q. Wang, H. Yin, T. Chen, J. Yu, A. Zhou, and X. Zhang, "Fast-adapting and privacy-preserving federated recommender system," *The VLDB Journal*, pp. 1-20, 2021.

[63] B. Alhijawi and Y. Kilani, "The recommender system: a survey," *International Journal of Advanced Intelligence Paradigms*, vol. 15, no. 3, pp. 229-251, 2020.

[64] M. H. Mohamed, M. H. Khafagy, and M. H. Ibrahim, "Recommender systems challenges and solutions survey," in *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*, 2019: IEEE, pp. 149-155.

[65] T. Maan, S. Gupta, and A. Mishra, "A Survey On Recommendation System," in *international conference on recent innovations in management, engineering, science and technology (RIMEST 2018)*, 2018, pp. 543-549.
[66] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 34-43, 2001.
[67] I. Tiddi, "Foundations of explainable knowledge-enabled systems," *Knowl. Graph. eXplainable Artif. Intell.: Found. Appl. Challenges*, vol. 47, p. 23, 2020.
[68] A. Velu and M. Thangavelu, "Hetero-GCD2RDF: An Interoperable Solution for Geospatial Climatic Data by Deploying Semantic Web Technologies," *Wireless Personal Communications*, vol. 117, no. 4, pp. 3527-3551, 2021.
[69] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data: The story so far," in *Semantic services, interoperability and web applications: emerging concepts*: IGI Global, 2011, pp. 205-227.
[70] M. Tamer Özsu, "A Survey of RDF Data Management Systems," *arXiv e-prints*, p. arXiv: 1601.00707, 2016.
[71] Š. Čebirić *et al.*, "Summarizing semantic graphs: a survey," *The VLDB journal*, vol. 28, no. 3, pp. 295-327, 2019.
[72] B. Makni, I. Abdelaziz, and J. Hendler, "Explainable deep RDFS reasoner," *arXiv preprint arXiv:2002.03514*, 2020.
[73] T. Berners-Lee, "Giant global graph," *Decentralized Information Group*, p. 29, 2007.
[74] D. Allemang, J. Hendler, and F. Gandon, *Semantic web for the working ontologist*. Elsevier, 2020.
[75] https://lod-cloud.net/ (accessed 10/2/2022.
[76] T. Di Noia *et al.*, "Building a relatedness graph from linked open data: A case study in the it domain," *Expert Systems with Applications*, vol. 44, pp. 354-366, 2016.
[77] M. Lnenicka and J. Komarkova, "Developing a government enterprise architecture framework to support the requirements of big and open linked data with the use of cloud computing," *International Journal of Information Management*, vol. 46, pp. 124-141, 2019.
[78] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The semantic web*: Springer, 2007, pp. 722-735.
[79] https://lod-cloud.net/versions/2007-05-01/lod-cloud.png (accessed 1/2/2022.
[80] M. Uschold, "Demystifying OWL for the Enterprise," *Synthesis Lectures on Semantic Web: Theory and Technology*, vol. 8, no. 1, pp. i-237, 2018.
[81] R. M. Nawi, S. A. M. Noah, and L. Q. Zakaria, "Issues and Challenges in the Extraction and Mapping of Linked Open Data Resources with Recommender Systems Datasets," *Journal of Information Science Theory and Practice*, vol. 9, no. 2, pp. 66-82, 2021.
[82] D. Brickley and L. Miller, "FOAF vocabulary specification 0.99. 2014," *Namespace Document. Available online: http://xmlns. com/foaf/spec/(accessed on 23 November 2018)*, 2015.
[83] N. Elahi, R. Karlsen, and E. J. Holsbø, "Personalized photo recommendation by leveraging user modeling on social network," in *Proceedings of International Conference on Information Integration and Web-based Applications & Services*, 2013, pp. 68-71.
[84] N. Chouchani and M. Abed, "Automatic generation of personalized applications based on social media," *Procedia Computer Science*, vol. 170, pp. 825-830, 2020.

[85] A. Tversky, "Features of similarity," *Psychological review*, vol. 84, no. 4, p. 327, 1977.
[86] A. Mahmoud and M. Zrigui, "Semantic similarity analysis for corpus development and paraphrase detection in arabic," *Int. Arab J. Inf. Technol.*, vol. 18, no. 1, pp. 1-7, 2021.
[87] P. Ristoski, J. Rosati, T. Di Noia, R. De Leone, and H. Paulheim, "RDF2Vec: RDF graph embeddings and their applications," *Semantic Web*, vol. 10, no. 4, pp. 721-752, 2019.
[88] K. W. Church, "Word2Vec," *Natural Language Engineering*, vol. 23, no. 1, pp. 155-162, 2017.
[89] T. Adewumi, F. Liwicki, and M. Liwicki, "Word2Vec: Optimal hyperparameters and their impact on natural language processing downstream tasks," *Open Computer Science*, vol. 12, no. 1, pp. 134-141, 2022.
[90] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111-3119.
[91] B. P. Chamberlain, E. Rossi, D. Shiebler, S. Sedhain, and M. M. Bronstein, "Tuning Word2vec for large scale recommendation systems," in *Fourteenth ACM Conference on Recommender Systems*, 2020, pp. 732-737.
[92] H. Caselles-Dupré, F. Lesaint, and J. Royo-Letelier, "Word2vec applied to recommendation: Hyperparameters matter," in *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018, pp. 352-356.
[93] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
[94] E. D. j. D. Team, "Deeplearning4j: Open-source distributed deep learning for the JVM," 2018.
[95] https://deeplearning4j.konduit.ai (accessed 2022).
[96] S. Alfarhood, *Exploiting Semantic Distance in Linked Open Data for Recommendation*. University of Arkansas, 2017.
[97] S. Zhang, Y. Ouyang, J. Ford, and F. Makedon, "Analysis of a low-dimensional linear model under recommendation attacks," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp. 517-524.
[98] S. Banerjee and K. Ramanathan, "Collaborative filtering on skewed datasets," in *Proceedings of the 17th international conference on World Wide Web*, 2008, pp. 1135-1136.
[99] Q. Ye *et al.*, "A unified drug–target interaction prediction framework based on knowledge graph and recommendation system," *Nature communications*, vol. 12, no. 1, pp. 1-12, 2021.
[100] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender systems handbook*: Springer, 2011, pp. 257-297.
[101] https://bnb.data.bl.uk/sparql (accessed 1/1/2022, 2022).

الخلاصة

الويب الدلالي والبيانات المفتوحة المرتبطة (LOD) جعلت البيانات قابلة للقراءة من قبل الأجهزة فضلاً عن المستخدمين. LOD ، هو مجموعة مبادئ نشر البيانات المهيكلة على الويب وربطها بعلاقات ، وذلك أتاح فرصاً للباحثين للاستفادة من البيانات الضخمة لتطوير تطبيقات في مجالات مختلفة. نظم التوصية التقليدية التي تعتمد على تقييمات المستخدمين للتوصية بالعناصر ذات العلاقة تعاني من مشكلة البداية الباردة Cold-start بينما الأنظمة الأخرى التي تعتمد على المحتوى تعاني من قلة المعنى الدلالي ,ومحدودية تحليل المحتوى. لذا فإن أنظمة التوصيحة تحتاج لإثراء العناصر والمستخدمين بخصائص وصفات إضافية واستغلال المعرفة الموجودة في LOD من أجل حل المشكلات الشائعة للأنظمة التقليدية وحساب المسافة الدلالية لإنتاج التوصيات.

تعتمد معظم الدراسات السابقة على مقاييس التشابه النقية القائمة على الارتباط ، والتي ستقلل من دقة التوصية إذا لم تكن هناك روابط كافية بين الموارد على الرغم من أنها تشترك في العديد من الصفات. في هذه الأطروحة ، يتم استغلال العلاقات المباشرة وغير المباشرة بين الموارد ويتم اقتراح نهج جديد لحساب تشابه الموارد بناءً على هذه العلاقات وإعطاء أهمية أكبر للروابط المباشرة باستخدام مقياس ELDSD المقترح ، بالإضافة إلى دمجها مع مقياس يعتمد على الميزات المستخرجة من مجموعتي بيانات بعد تمثيل هذه الميزات باستخدام نموذج Word2Vec للتوصية بالأفلام ذات الصلة. بالإضافة إلى التوصية بعناصر من مجالات أخرى ذات علاقة بالأفلام الموصى بها مثل الموسيقى والكتب ذات الصلة.

تم إجراء تقييم تجريبي باستخدام مجموعة بيانات أفلام MovieLens و مجموعة بيانات أفلام Yahoo وتشير النتائج إلى أن النهج المقترح يتفوق في الأداء على الأساليب الأخرى التي تعتمد على هيكلية الروابط بما يصل إلى 7.8% تحسين في مقياس F وبما يصل إلى 31.8% على الطرق الأخرى التي تعتمد على خصائص العناصر فقط.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل / كلية تكنولوجيا المعلومات
قسم البرامجيات

تحسين نظام توصية مستند على البيانات المفتوحة المرتبطة لنطاقات متعددة

اطروحة مقدمة
الى مجلس كلية تكنولوجيا المعلومات - جامعة بابل وهي جزء من متطلبات
نيل درجة الدكتوراه فلسفة في تكنولوجيا المعلومات / برمجيات

من قبل
أحمد مناف مهدي محمد

بإشراف
الأستاذ الدكتور
أسعـــد صبـــاح هـــادي عبـاس

1444هـ 2022م