Universidad Nacional de Educación a Distancia

Master Thesis

Automatic Generation of Domain Knowledge Graph for Recommendation Systems using Open Source Resources

Author:
Almudena Sanz Olivé

Supervisors:
Jorge Carrillo de Albornoz Cuadrado
Miguel González-Fierro

Master Universitario en Lenguajes y Sistemas Informáticos

September 2020

Abstract

A group of state-of-the-art recommendation algorithms using Knowledge Graphs are RippleNet [1] or KGAT [2]. However, the main bottleneck to benchmark and work with these algorithms is that they used Microsoft Satori [1] or Freebase [2] (now Google Knowledge Graph [3]) as their KG. As Satori and Google Knowledge Graph are commercial KGs and not open source, it's not possible to replicate the results found in the papers on new data, or use these solutions in real applications without paying for access.

This limitation is critical in the development of this area of RS. There are researchers working on algorithms and applications, whose results cannot be replicated openly by the science community, hence, going against the scientific method. As researchers, one of our main goals is to make science accessible and replicable for all the scientific community, empowering the development of new knowledge areas.

To fill this gap, this Thesis presents a system to generate domain adapted Knowledge Graphs using open source information from Wikidata. These Knowledge Graphs can be used in hybrid Recommendation Systems, that use linked knowledge on the items as side information, combining both Collaborative Filtering and Content Base Filtering strategies.

The results show that the proposed system is able to create domain adapted Knowledge Graphs from open source information for recommendation datasets, and that the KGs generated are able to compete with their commercial versions. We have shown that our proposed system creates smaller KGs that are more domain adapted, that have a similar efficiency in downstream tasks of Recommendation Systems than commercial KGs. This could be specially relevant in systems that require faster computational times that can be achieved with smaller KGs, as real-time systems, or to save computational cost in high-scale systems. The system ca be used as a reference to evaluate the state-of-the-art algorithms in future works in the area.

[1]https://searchengineland.com/bings-knowledge-repository-satori-just-got-a-lot-smarter-179800
[2]https://developers.google.com/freebase
[3]https://developers.google.com/knowledge-graph

Acknowledgements

I would like to express my sincere gratitude to every person who has helped me in my path to finish successfully my Masters Degree.

To my Thesis's director, Jorge, who has helped me understand and organize the topic of the Thesis. He has also guided my in the research process, and has encouraged and motivated me to look further into the problem and expand my research.

To Miguel, my Thesis's co-director, who helped me find the topic of research and supported me. To the Microsoft team, who have helped me in the process of doing a collaboration research project, allowing me to participate in their open-source Recommendation repository.

To all my professors in UNED. They have helped me enter and navigate the topic of Computational Languages, opening a new area of interest.

To my friends, who have helped me through the bad and the good times, and who have always believed in me and tried to understand me. Than you for cheering for me.

To my family, for their infinite support and comprehension. None of this would have been possible without you.

——

Me gustaría expresar mi más sincera gratitud a todas las personas que me han ayudado en el camino a realizar mi Máster en Lenguajes y Sistemas Informáticos.

A mi director de Tesis, Jorge, que me ha ayudado a entender y organizar el tema del trabajo. También me ha guiado en el proceso de investigación, y me ha animado y motivado a mirar más allá en mi problema de investigación y a expandir mi trabajo.

A mi co-director de Tesis, Miguel, que me ha ayudó a encontrar el tema de mi investigación y me ha apoyado de manera continuada. Al equipo de Microsoft, que me han ayudado en el proceso de hacer un proyecto de colaboración, permitiéndome participar en su repositorio open-source de Recomendación.

A todos mis profesores en la UNED. Me han ayudado a entrar y navegar el mundo de Lenguajes Informáticos, abriendo para mi un nuevo área de interés.

A mis amigos, que me han ayudado en lo bueno y en lo malo, y que siempre han creído en mi y han tratado de entenderme. Muchas gracias por los ánimos.

A mi familia, por su apoyo y comprensión infinitas. Nada de esto habría sido posible sin vosotros.

Contents

List of Figures

List of Tables

Acronyms

KG	Knowledge Graph
RS	Recommendation System
KGAT	Knowledge Graph Attention Network
CF	Collaborative Filtering
ACF	Automated Collaborative Filtering
CBF	Content-Based Filtering
NN	Neural Networks
KGE	Knowledge Graph Embedding
CTR	Click Through Rate
ID	unique identifier

Dedicado a mis abuelos, que siempre alentaron mi curiosidad

To my grandparents, who always encouraged my curiosity

Chapter 1

Introduction

We live in the information era and our principal way of interacting with the world is through digital platforms. Every day we use digital apps to communicate with our friends, colleagues and family. We use social networks to get information about what is happening in the world, we consult webpages to investigate about specific topics, we use content platforms to consume media content, we browse and shop in different retailer webpages or we consult our finances using digital apps. All of our digital interactions with the world are recorded and stored by the different services we use, leaving a digital trace.

Since new information is constantly being created and data is evolving, there is a risk of being overwhelmed or lost when dealing with all of this information. We do not have time to check all of the news, browse all of the new products available in an online store, or check all of the new media available for our entertainment. On the other hand, it is critical to the services giving us this information that what they show to the users will finally be consumed, otherwise it would be a waste of resources and probably will reduce their engagement. That is why it is important that the systems that provide us with information become more intelligent in trying to give us the specific pieces that can be relevant to us. Providing services with this type of intelligence is a problem covered by the area of Recommender Systems (RS). These systems are defined to find the intersection between two sets of information, the set of information that will be relevant to us and the set of information that is available to be consumed.

The set of information that is relevant to us is stored implicitly in the interactions with the system that we have had in the past, our digital trace. If we are interested in music and we consume music content, it is probable that we will be interested in consuming more music content in the future. However, if we listen to a song and the system keeps recommending only that specific song, probably we will not want to only listen to it again and again. Hence, the objective of the RS is to show us other new songs that would be interesting to us. The RS makes use of the set of information that is available to be consumed to recommend us other items based on our preferences. As we have mentioned, this set of available information to be consumed can be overwhelmingly big, and needs to be prioritised and filtered given our preferences.

Hence, RS are designed to find a specific set of items that will be of interest for a specific user. There are different types of RS based on the information they use to provide recommendation to users, defined as recommendation strategies. The most popular types are:

- Collaborative filtering (CF): recommendation items for each active user are generated using the items that other users with similar preferences liked in the past.
- Content Based Filtering (CBF): recommendations depends on the content of the items selected by the users in the past.
- Hybrid Filtering mix techniques of both methods mentioned above.

Collaborative filtering (CF) considers the historical interactions of a set of users with the available items, and uses the potential common preferences between users to make further recommendations for a user with items they have not explored. However, the main problems found with this approach are the sparsity of user-item interactions and the cold start problem. Normally, the set of available information to be consumed is very large compared to the set of items that a specific user has interacted with, if we represented the information of the interactions as a matrix of user-items, we would find a very sparse matrix, with many 0s and a few 1s. Also, when a user has just started interacting with the system, there are very few records of their interests, what we call the cold start problem.

Content Based Filtering models can only be as complex as the content to which it has access, and the content of the items needs to be extracted in a structured way. For instance, in the case of movies, if the only available information on movies is the genre, this is the only dimension the model can incorporate.

A branch of research on RS has focused on incorporating side information to address the above mentioned problems, this side information consists of content for the items consumed. Thus, this branch of RS combine CF and CBF into a hybrid approach.

One of the proposed solutions to this problem is to use a Knowledge Graph (KG), a graph that maps the relationships between entities. KGs are made of triple facts, sets of <e1, rel, e2>, being *e*1 and *e*2 the entities related and *rel* the type of relationship between the entities. As seen in the figure below, in a KG an entity can be related to many other entities through different connections. The figure below shows the movie *Forest Gump* is related to *drama* by the relationship *genre* and to *Tom Hanks* by the relationship *star*.

KGs enrich the user-item interactions by adding extra information on how items are related.

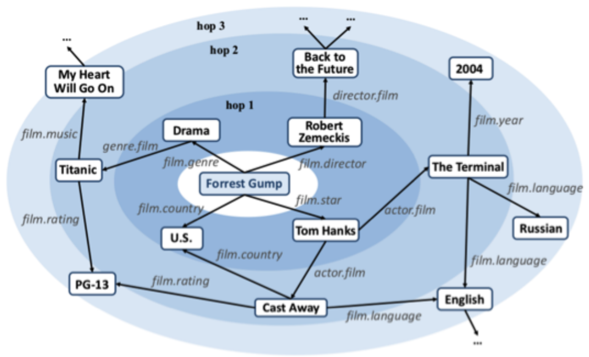


FIGURE 1.1: Hops in a knowledge Graph. Retrieved from Wang et al. [1].

This approach addresses the problem of sparsity of user-item interactions by creating a distance measure of the relationships between items, since we would not be looking at just a matrix of interactions with isolated items because the items would have a level of relationship between them. This distance measure helps us prioritise items that are close in the KG. If a user has seen Wall-e and the movie has been created by Pixar, the KG provides the information of other movies created by Pixar and those will probably be relevant to the user. The KG used must contain the items of the recommendation dataset, together with their related triples.

KGs can benefit Recommendation systems in 3 ways [1]:

1. Improving the accuracy of the models. By adding side information to CF using KGs, the systems are able to address sparsity and cold-start problems. As further explained below, the algorithms using this strategy improve the accuracy of state of the art solutions.

2. Improving the reasoning of the model as providing a reason why a recommendation was made. The relationship between the items of the KG can provide explainability on how items are connected and why a new item is relevant. For example, if the user has watched the movie *Forrest Gump*, and the recommender suggests *The Terminal*, using the KG it can be inferred that the connection between those two items is that they star the actor *Tom Hanks* and use it as an explanation to the user.

3. Improving the diversity of the results by introducing new relationships between items that may not be present in the user-item interaction. If a movie is new and it does not have historical rating information, it will not be prioritized by a CF recommender as it is not present in the user-item ranking. However this can be solved by introducing the KG as content information into a hybrid recommender. If *Tom Hanks* stars in a new movie, it will appear as a new triple in the KG and that movie will be present in the set of available information and prioritized based on its position in the KG.

A group of state-of-the-art recommendation algorithms using Knowledge Graphs are RippleNet [1] or KGAT [2]. They use a combination of path based and embedding based methods on the KGs and are able to improve state-of-the-art metrics on both ranking and rating recommendation problems. The results of their respective papers show that KGAT beats state-of-the-art recommendation algorithms on ranking evaluation for Amazon Book Review [6], Last-FM and Yelp2018 datasets. RippleNet beats state-of-the-art solutions on rating evaluation for the datasets Movielens 1M [5], Book-Crossing and Bing-News. Both provide access the code in their respective Github repositories.

However, the main bottleneck to benchmark and work with these algorithms is that they used Microsoft Satori [1] or Freebase [2] (now Google Knowledge Graph [3]) as their KG.

[1] https://searchengineland.com/bings-knowledge-repository-satori-just-got-a-lot-smarter-179800
[2] https://developers.google.com/freebase
[3] https://developers.google.com/knowledge-graph

As Satori and Google Knowledge Graph are commercial KGs not open source, it's not possible to replicate the results found in the papers on new data, or use these solutions in real applications without paying for access.

This limitation is critical in the development of this area of RS. There are researchers working on algorithms and applications, whose results cannot be replicated openly by the science community, hence, going against the scientific method, and this is a difficult challenge to do further research in the area. The only method to replicate or advance on them is by using private resources from specific companies that will ultimately have control over the advance. As researchers, one of our main goals is to make science accessible and replicable for all the scientific community, empowering the development of new knowledge areas.

Exploring the available KGs, we have found that the most well-known are proprietary, such as the mentioned Microsoft Satori or Google Knowledge Graph. These KGs are built and maintained by Microsoft and Google respectively, and they require a payment for the use of their API. Other available KGs can be used openly for commercial use, such as Yago and DBPedia. These open KGs cover many domains and are either extracted from Wikipedia, or built by communities of volunteers [3].

However, prominent open source KGs, such as DBpedia and YAGO focus on extraction from info-box tables in Wikipedia, so the information available can be scarce with this restriction. Also, finding specific items in those KGs can be a challenge, since the items are stored using specific ids that need to be used to extract the items. For example, in DBpedia, to extract the information on the movie *Forrest Gump* the query needs to link to the specific page `http://dbpedia.org/page/Forrest_Gump`.

Hence, it would be very useful if a system could create KGs that are open and reusable, and even more if these KGs could be adapted to the domain of a specific recommendation dataset, limiting their size and thus the computational costs to use them.

To address the problem of building an open source KG that can be used for RS, we have designed a system that uses Wikidata, the free and open knowledge base of Wikipedia. This database is constantly being updated by the new articles and contributions of users. The Wikidata API provides more structured information than DBPedia and

YAGO, because these two KGs are restricted to Wikipedia infoboxes, inboxes are fixed-format tables usually added to the top right-hand corner of pages to consistently present a summary. However, Wikidata acts as central storage for the structured data of the Wikimedia ecosystem [4], and contains all the links of Wikipedia pages, having much more structured information than the infoboxes. Also, the data in Wikidata is published under the Creative Commons Public Domain Dedication 1.0, allowing the reuse of the data in many different scenarios. Users can copy, modify, distribute and perform the data, even for commercial purposes, without asking for permission.

Wikidata stores records of the Wikipedia pages, with specific characteristics of each page and also links to the related Wikidata entries. Wikidata also uses specific item ids, for example, the movie *Forrest Gump* is represented by the item ID *Q134773* https://www.wikidata.org/wiki/Q134773. However this challenge is solved by using the Wikipedia search API, that receives queries and retrieves the top matching Wikipedia page results from which the Wikidata item id can be extracted.

Wikidata uses specific terms for the representation of KG triples. A triple is defined as a statement, they are built by pairing a relationship, in Wikidata called property, with at least another item in Wikidata. An item by nature can have a relationship with multiple items, like the actors and actresses starring a movie. The proposed system is able to extract all triples linked to a specific item from Wikidata, hence building a sub-graph od the items linked to an original item.

The result of this design is a system that can receive a list of items from a specific domain or RS dataset, disambiguate them and find the specific Wikidata ID, and retrieve the related items from the Wikidata database, building a domain adapted KG.

The resulting KG generated by the system can be reused in any hybrid RS. Hence, state-of-the-art hybrid models can be compared with any other hybrid RS. Thus, the system generates KGs that serve as a reference framework.

[4] https://meta.wikimedia.org/wiki/Wikidata/Notes/DBpedia_and_Wikidata

1.1 Proposal and Objectives

1.1.1 Proposal

The goal of this master thesis is to address the mentioned gap, building a flexible module to create domain adapted KGs from open sources and use them in state-of-the-art recommendation algorithms, evaluating their performance against commercial KGs.

The designed system will receive as an input a RS dataset consisting of of item reviews (movies, books...), and will output a domain adapted Knowledge Graph, a directed graph with the item attributes and relations between the items of the dataset. For example, if the input is a dataset of movie reviews, the system will build a KG in which each movie will be linked to its director, cast, awards and genres, those movies sharing a director, cast member, award or genre will be also linked.

The KG is defined as a directed graph composed of subject-property-object triple facts. These triple facts will be extracted from external sources of information, the open source databases Wikipedia and Wikidata. The KGs generated by the system will be domain adapted because it will be composed of the relations shared between the items of the input datasets.

First each item name has to be disambiguated, as an example, for the movie title ”The Godfather”, the system needs to find the Wikidata entry that corresponds to the movie instead of the book. Once the Wikidata entry is located, the system extracts all relations from the initial item to other Wikidata entries, for example the director, cast or any other listed properties.

This output Knowledge Graph will then be used downstream as auxiliary data in two state-of-the-art Recommendation System: RippleNet [1] and KGAT [4]. The performance of rating and ranking based recommendation approaches of two state-of-the-art datasets algorithms RippleNet and KGAT is to be evaluated on Movielens 1M [5] and Amazon Book Review [6] respectively.

In the Evaluation Chapter we perform two types of Experiments: intrinsic evaluation of the KGs and extrinsic evaluation.

- In the intrinsic evaluation, the system will receive an input of a recommendation dataset, and will generate a domain adapted KG for the specific dataset using Wikidata. The generated KGs are compared to their equivalent commercial KGs, using the dataset Movielens 1M with Microsoft Satori and Amazon Book Review with Freebase, respectively. The evaluation will compare the completeness of the KG with respect to the number of items in the original dataset and the extension of the KG in number of triples retrieved. The size and completeness of the KGs generated by the system will be compared against the proposed commercial KGs Microsoft Satori and Freebase.
- In the extrinsic evaluation the generated KGs will serve as downstream inputs for the two mentioned hybrid Recommendation Systems. The output generated by the proposed system will be adapted to the format of the files expected by the Recommendation Systems. The algorithms will be trained and tested on the reviews dataset, and the performance of the recommendation of the test set will be measured. The results of the recommendations in terms on rating and ranking based evaluation are compared with the original results in the published papers, that used commercial KGs.

Results show that the performance of the domain adapted KGs generated by the proposed system in the two Recommendation Systems is at the same level of performance as the original for all of the metrics for RippleNet and for most of the metrics for KGAT. Hence, we are able to replicate the recommendation results using a non-commercial, open source, domain adapted KG. We have shown that our proposed system creates smaller KGs that are more domain adapted, that have a similar efficiency in downstream tasks than commercial KGs. This could be specially relevant in systems that require faster computational times that can be achieved with smaller KGs, as real-time systems, or to save computational cost in high-scale systems.

The final proposed solution serves as a base standard of methodology and architecture to generate new domain adapted Knowledge Graphs and that are open source. This means that the system is able to generate KGs from lists of items in any recommendation dataset, such as movies, books, people or entities, and it adds a disambiguation module for items that may be ambiguous. Furthermore, this thesis proposes a methodology, framework and system that can be freely reproduced and used in any hybrid Recommendation System using Knowledge Graphs as side information, achieving similar results to using commercial KGs.

Currently, there is no proposed solution in the literature to generate domain adapted KGs from recommendation datasets using open-source and up-to-date information. The closest match is KB4Rec [7], a linked dataset for research on knowledge-aware recommender systems. KB4Rec maps items into Freebase entities via title matching if there is a mapping available. However, these are pre-created datasets, not a framework to freely generate domain adapted KGs on open source information, and use Freebase owned by Google.

This solution will allow other researchers to develop further this research area without having to depend on commercial KGs. As mentioned, the ability to generate open source domain adapted Knowledge Graphs that can compete with commercial KGs is critical in the development of the field of Recommendation Systems with hybrid approaches, without having to depend on commercial information owned by specific companies. This is a step further to make science accessible and replicable for all the scientific community, empowering the development of new knowledge areas.

1.1.2 Objectives

The steps to achieve this goal are the following:

To analyse the existing open source KGs and evaluate how to adapt them to Recommendation Systems.

To build a system that can generate a domain adapted KG from a recommendation system review dataset.

To build a system that can process a specific list of items in order to find the neighbours of each item by using open source KGs. This will be achieved applying web mining and graph mining techniques on Wikidata and the Wikimedia ecosystem.

To build a disambiguation module that can deal with selecting the correct item from the string name of the original item.

The proposed system will be evaluated intrinsically, generating domain adapted KGs for the datasets Movielens 1M and Amazon Book Review. The size and extension of the generated KGs will be compared to their commercial versions. The objective is to obtain metrics to measure if our system is able to cover the items contained in the dataset, evaluating if the disambiguation module is able to find the specific pages of the movies or books in the Wikidata database. The extension of the KG will be measured by the number of triples retrieved, evaluating if the extension of the domain of the KG is bigger or smaller to the extension of the commercial counterpart.

The generated KGs will be evaluated extrinsically, by using them as side information downstream in two Recommendation System models, replicating the results of the papers Ripple and KGAT on the datasets of Movielens and Amazon Book Review respectively. Results of using rating-based and ranking-based user models will be compared to evaluate if the generated domain adapted KGs are suited to be used as side information in hybrid RS and the performance obtained with them is at the same level as the performance obtained with their commercial counterparts.

The code generated for the above mentioned goals will contribute to the open source Recommender repository from Microsoft and will be evaluated by a group of Data Scientist and Scientist from Microsoft. It will also be freely available in my personal Github account, under the GPLv3 license, referenced and explained in the Appendix A.

1.1.3 Organization, structure of the document

This document is organized as follows.

- Chapter 1 Introduction
- Chapter 2 State of the Art: presents an overview of the methods an approaches used in related work and introduces the formal definition of the problem and the machine learning algorithms used to address it
- Chapter 3 Automatic Generation of Domain Knowledge Graph for Recommendation Systems: describes proposed solution as the software architecture
- Chapter 4 Evaluation and Discussion: evaluates the results of the testing of the proposed solution with real data in different experiments and discusses the results obtained
- Chapter 5 Conclusions and Future Work: summarizes the main contributions and describes future work and opened issues worth studying

Chapter 2

State of the Art

2.1 Recommendation Systems

Recommendation Systems (RSs) have become very popular in recent years and are used in various web applications, such as social media apps, media content apps, news portals or e-commerces. RSs are software tools that are used to provide suggestions to users [8]. These suggestions associate with various decision-making processes, when the available options are very extense and the user can benefit from suggestions to make the most appropriate choice, such as which items to watch, which items to buy or which items to read. *Item* is the general term used to denote what the system recommends to users.

Recommendation is based on the problem of estimating the ratings an user will assign to specific items. A rating is a measure of how much an user likes or dislikes an item. Once the system estimates ratings for all items, RSs recommend to the user the item(s) with the highest estimated rating(s).

It can be argued that the Grundy system [9] was the first recommender system. The system proposed using stereotypes or *models of users*, called the User Synopsis or USS, based on the user's demographics and their responses to specific preference questions, such as if they like mystery books. Building manually stereotypes, and associating each individual user to one of the stereotypes, the Grundy system recommends relevant books to each user.

RSs emerged as an independent research area in the mid 1990's, when researchers started focusing on recommendation problems that explicitly rely on the ratings structure [10] using Collaborative-filtering methods, that will be explained below.

There are different types of Recommendation Systems, based on the information they use to provide recommendation to users [11]:

- **Collaborative-filtering (CF)**: CF is the most extensively used approach to design recommender systems. The recommendation items for each active user are generated using the items of other users with similar preferences liked in the past [10]. The first papers on CF appeared in the mid-1990s, such as Hill, Will, et al. [12] video recommendation or GroupLens [13] in 1994, a system for collaborative filtering of netnews, to help people find articles they will like in the huge stream of available articles. CF systems all follow similar processes of gathering ratings from users, computing the correlations between pairs of users to identify those users with similar tastes, and combining the ratings of those similar users to make recommendations. They will be explained in detail in Section 2.1.1.
- **Content-Based filtering (CBF)**: in CBF recommendations depends on the content of the items selected by the users in the past, recommending the best-matching item to the user's past preferences. The content-based approach to recommendation has its roots in information retrieval [14] and information filtering research [15]. An example of content-based recommendation is the Fab system [16], which recommends Web pages to users, the content of each of the web pages is represented by the 100 most important words of each webpage, and the similarity between pages is measured based on the overlap of most important words. Users received recommendation of webpages with high similarity to those that they had already selected in the past.
- **Hybrid filtering**: The hybrid filtering is a combination of more than one filtering approach, such as collaborative and content-based filtering. The hybrid filtering approach is introduced to overcome some common problems that are associated with above filtering approaches such as cold start problem, overspecialization problem and sparsity problem. This can be achieved by introducing in CF approaches content information, such as the contextual reviews, relational data and Knowledge Graphs [17]. The implementation of hybrid filtering can also improve the

accuracy, diversity, explainability and efficiency of recommendation process [1] by introducing side information on the items that will contain new relationships between items that may not be present in the user-item interaction, enriching the inference of similarities between items. These new introduced relationship can help explaining how items are connected when making the recommendation, (eg. *this movie is recommended because you liked this other movie*).

There are other types of recommendation such as demographic filtering, in which recommendations are established on a demographic profile of the user. The recommendations are based on the information provided by the user. Users are considered to be similar according to demographic parameters such as nationality, age, gender, etc. [8]. However, this type of models constitute a small portion of RS. This thesis will focus on CF, CBF and hybrid systems.

2.1.1 Collaborative Filtering

Collaborative Filtering (CF) is the process of filtering or evaluating items using the opinions of other people. CF uses the assumption that people with similar tastes will rate things similarly. While the term collaborative filtering has only been around for a little more than a decade, CF is based on a basic human behaviour: sharing our opinions with others.

As a formal area of research, CF started as a solution to finding relevant information in growing text repositories. The content of text databases such as libraries databases, corporate document sets, e-mail archives and forums started growing exponentially, and it was more and more difficult to find items that were relevant. Pure content-based techniques, based on keyword matching or content matching, were often inadequate at helping users find the documents they wanted as they did not incorporate information on the relevance or quality of the content, something implicit in user ratings [18]. Hence, CF is a very powerful technique to infer relevance of items without having information on the content or features of the item per se, as the recommendation is purely based in user ratings. This makes the method very general for any item recommendation, since the nature of the item does not really matter.

The Tapestry system [19], developed at Xerox PARC, took the first step towards incorporating user actions and opinions into a message database and search system. Tapestry stored the contents of messages, along with metadata about authors, readers, and responders. It also allowed any user to store annotations about messages, such as "*very useful survey*" or "*Sara should read it!*". Tapestry users could use queries that combined basic textual information (e.g. contains the phrase "recommender systems") with semantic metadata queries (e.g. written by Sara OR replied to by Javier) and annotation queries (e.g. marked as "great" by Cristina) [20].

A limitation of active CF systems is that they require a community of people who know each other or can be in contact. To overcome this issue, Automated Collaborative Filtering (ACF) systems use a database of historical user opinions that automatically match each individual to others with similar opinions. The first ACF systems included GroupLens [13] for news articles, Ringo [21] for music and musical artists, and Bellcore's Video Recommendation [12] for movies. Bellcore project designed a video recommendation system based on the premise " *When making a choice in the absence of decisive first-hand knowledge, choosing as other like-minded, similarly-situated people have successfully chosen in the past is a good strategy*". The system used email to ask the user to evaluate 500 videos on a scale of 1-10 (1 being low and 10 high) for the titles they have seen. They rate unseen movies as *must-see* or *not-interested.* The system looks for correlations between the user's ratings and ratings from a random subsample of known users. The most similar users found are used as variables in a multiple-regression equation to predict the new user's ratings. The multiple-regression equation is evaluated by a hold-out of movies for testing. The system used a random subsample to limit the number of correlations computed to be O(n) rather than O(n^2) in the number of participants.

These ACF systems all follow similar processes of gathering ratings from users, computing the correlations between pairs of users to identify a those users with similar tastes, and combining the ratings of those similar users to make recommendations.

The main problems found with this approach are the sparsity of user-item interactions and the cold start problem. Normally, the set of available information to be consumed is very large compared to the set of items that a specific user has interacted with, if we represented the information of the interactions as a matrix of user-items, we would find a very sparse matrix, with many 0s and a few 1s, hence computing the correlation or similarity between pairs of users can be hard. Also, when an user has just started interacting with the system, there is very little information to generate accurate recommendations, what we call the cold start problem.

Another problem implicit in this method is the limitation of the number of correlations or similarities between users. The order or magnitude of the computation is O(n2) for the number of participants, hence growing exponentially with the number of users if the system needs to compute similarities one-to-one.

2.1.2 Content-based filtering

Content-based filtering (CBF) uses the assumption that items that are similar, with similar features, will be rated similarly. For example, if an user liked a web page of the movie *Wall-E*, the assumption is that that user will also like another web page that has content about *Wall-E*. The challenge for this systems is to extract the features of items that are most predictive. User profiles are built of features from the items they have rated, then that user profile is compared to item features of new items, those items with matching features to the users preferences are ranked higher. Features can be available characteristics of the items, for example, in movies it can be the director, genre, starring actors, content of the plot, awards... As mentioned earlier, in the Fab system [16], which recommends Web pages to users, the content of each of the web pages is represented by the 100 most important words of each webpage. These features need to be extracted for all items.

Content-based filtering can recommend items without previous user ratings. For example, if the user profile indicates they like movies starred by Tom Hanks, and a new movie starred by Tom Hanks comes out, the features of the movie will match the user's profile and it will be recommended, even if the new movie does not yet have user ratings. This is especially important in applications where new items are generated everyday, like news. However, CF needs ratings from users in order to make a recommendation.

CBF models can only be as complex as the content to which it has access. For instance, in the case of movies, if the only available information on movies is the genre, this is the only dimension the model can incorporate. Furthermore, if there is no easy way to automatically extract a feature, then that feature cannot be incorporated into the model, because it would not be scalable. For example, while people find the quality of multimedia data (e.g., images, video, or audio) for web pages important, it is difficult to automatically extract this information.

One of the drawbacks of CBF is that it may over-specialize. Items that match the content features in the user's interest profile are recommended, whereas items that do not contain the exact features specified in the user's interest profile may not get recommended even if they are similar (e.g., due to synonymy in keyword terms).

2.1.3 Hybrid Systems

Content-based filtering (CBF) and Collaborative filtering (CF) systems may be combined to address each others weaknesses.

CF allows evaluation of the relevance or quality of the items, because people rating the items are implicitly making this evaluation [8], whereas for CBF it is difficult to automatically extract this information.

As mentioned, one of the drawbacks of CBF is that it may over-specialize. Researchers generally believe CF leads to more unexpected or different items that are equally valuable. Some people call this property of recommendations novelty or serendipity [22],

This combination of CF and CBF is defined as hybrid systems. One option is to perform the combination manually by the end-user specifying particular features, essentially constraining recommendations to have certain content features. More often they are automatically combined, sometimes called a hybrid approach. There are many ways to combine them, and no consensus exists among researchers [8].

Hybrid recommender systems, which combine collaborative filtering and auxiliary information such as item content, can usually achieve better recommendation results and have gained increasing popularity in recent years. [23]

Combining CF with Content-based methods, hybrid RS deal with these issues by introducing various side information on items, such as the contextual reviews [24], relational data [25], such as social networks, and Knowledge Graphs [26]. Another advantage using side informations, is the explainability of the results, being able to explain why an item is being recommended out of others. This has been found to be important for the effectiveness, efficiency, persuasiveness, and user satisfaction of recommender systems [27]. Structural knowledge has shown great potential in providing rich information about the items of RS, offering a promising solution to improving the accuracy and explainability of recommendations.

2.2 Knowledge Graphs in Hybrid Recommendation Systems

In particular, Knowledge Graphs show great potential as being used as side information on hybrid recommendation systems due to its well defined structures resources. This type of methods mostly transfer structural knowledge of entities from KG to user-item interaction modelling based on the given mapping between entities and items [17], as pictured in the figure below. Since KGs provide rich information including both structured and unstructured data with different semantics, the usage of the knowledge base within the context of hybrid recommender systems are attracting increasing attention. [28].

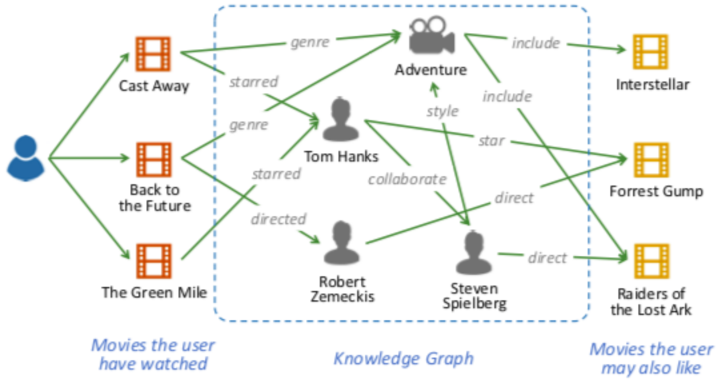


FIGURE 2.1: Interaction of KGs in Hybrid Recommendation [1]

2.2.1 Knowledge Graphs

The foundations of Knowledge Graphs trace back to the 1950s, when Semantic Networks were invented to address the growing need for a knowledge representation framework that can capture a wide range of entities (i.e. real-world objects), events, situations or abstract concepts and relations. The end application was to extended English Dialogue tasks. The main idea behind Semantic Networks was to capture a wide range of issues which includes the representation of plans, actions, time, individuals' beliefs and intentions, and be general enough to accommodate each issue [29]. Semantic Networks are a general graphic notation to represent knowledge in patterns of interconnected nodes. *Nude* [30], the first semantic network, was created by R. H. Richens of the Cambridge Language Research Unit in 1956 as an interlingua for machine translation of natural languages [31]. The idea behind it was that instead of using direct translation, say from Spanish to English, the system would translate first to a neutral conceptual language or interlingua, and from that to the final language. They explained: "*The elements represent things, qualities or relations... A bond points from a thing to its qualities or relations, or from a quality or relation to a further qualification* [30]".

Knowledge Graph theory was initiated by C. Hoede, a discrete mathematician at the University of Twente, and F.N. Stokman, a mathematical sociologist at the University of Groningen [32]. The initial idea was to use graphs, a discrete mathematical concept, as a representation of the contents of medical and sociological texts. The objective was to use graphs in such a way that the resulting structure could function as an expert system, could automatically search for causes, and could act as a decision support system in medicine with an automated calculation of consequences. This put the focus on causal relationships between medical concepts. These relationships were extracted from English texts searching for linguistic indicators like *A "causes" B* or *C "by" D*. Hence, a KG was a type of Semantic Network, but with some added restrictions, like the type or relationships included in the graph, to facilitate algebraic operations on the graph.

There is no formal definition of Knowledge Graph. In a broader perspective, a Knowledge Graph is a variant of Semantic Network with added constraints whose scope, structure, characteristics and even uses are not fully realized and in the process of development. There is a minimum set of characteristics of knowledge graphs, used to differentiate KGs from other sets of information. A KG has the following characteristics [33]:

- Mainly describes real world entities and their interrelations and are organized in a graph.
- Defines possible classes and relations of entities in a schema.

- KGs are made of triple facts, sets of <e1, rel, e2>, being $e1$ and $e2$ the entities related and rel the type of relationship between the entities.
- Allows for potentially interrelating arbitrary entities with each other.
- Covers various topical domains.

For example, Knowledge Graphs on the Semantic Web are typically provided using Linked Data as a standard. They can be built using different methods: they can be curated by an organization or a small, closed group of people, crowd-sourced by a large, open group of individuals, or created with heuristic, automatic or semi-automatic means.

Since proposed, KG have attracted much attention in many fields, ranging from recommendation, dialogue system, to information extraction [17].

2.2.2 Methods to include KGs in hybrid RS

Hybrid recommender systems can benefit from KGs in the following ways:

- Improving the accuracy of the models, by adding side information to CF using KGs. The systems are able to address sparsity and cold-start problems. As further explained below, the algorithms using this strategy improve the accuracy of state of the art solutions.
- Improving the reasoning of the model as providing a reason why a recommendation was made. The relationship between the items of the KG can provide explainability on how items are connected and why a new item is relevant. For example, if the user has watched the movie *Forrest Gump*, and the recommender suggests *The Terminal*, using the KG it can be inferred that the connection between those two items is that they star the actor *Tom Hanks* and use it as an explanation to the user.
- Improving the diversity of the results by introducing new relationships between items that may not be present in the user-item interaction. If a movie is new and it does not have historical rating information, it will not be prioritized by a CF recommender as it is not present in the user-item ranking. However, introducing

the KG as content information into a hybrid recommender, if *Tom Hanks* stars in a new movie, it will appear as a new triple in the KG and that movie will be present in the set of available information and prioritized based on its position in the KG and the user's preferences.

Hybrid Recommender Systems using Knowledge Graphs can be classified into three groups:

- Path Based Methods: methods that augment the data of user-item pairs with KG triplets.
- Embedding Based Methods: methods combining item and entity embeddings learned from different sources.
- Combination of Path and Embedding Based methods.

2.2.3 Path Based methods

Path Based methods explore the various patterns of connections among items in KG to provide additional guidance for recommendations. For example, Personalized Entity Recommendation (PER) [34] and Meta-Graph Based Recommendation [35] treat the KG as a heterogeneous information network, and extract meta-path/meta-graph based latent features to represent the connectivity between users and items along different types of relation paths/graphs.

Path-based methods rely heavily on manually designed meta-paths, which is hard to optimize in practice, as it which requires rich domain knowledge that is extremely difficult to obtain in complex, large-scale, and schema-rich KGs. A meta-path is a relation sequence connecting object pairs, and is widely used to extract structural features that capture relevant semantics for recommendation. Instead of modelling the two-way interaction e_1 <> e_2, path pased methods aim to characterize a three-way interaction e_1 *<meta-path>* e_2. To apply this methods to domain-adapted recommendation, the meta-paths related to the domain have to be designed manually, such as defining co-starring in a movie as the meta-paths: *"actor <movie> actor"* path, *"actor <movie-director-movie> actor"*, or explicit user-item meta-paths of the user-item interactions, such as *"user <movie1-director> movie2"*.

Another concern is that it is impossible to design hand-crafted meta-paths in certain scenarios (e.g., news recommendation) where entities and relations are not within one domain [1].

2.2.4 Embedding-based methods

Embedding-based methods pre-process a KG with knowledge graph embedding (KGE) algorithms and incorporate the learned entity embeddings into a recommendation framework form the items in the recommendation dataset. Embedding components automatically extract items' semantic representations from the Knowledge Graphs's structural content.

As examples, Deep Knowledge-aware Network (DKN) [36] treats entity embeddings and word embeddings as different channels, then designs a CNN framework to combine them together for news recommendation, fusing semantic-level and knowledge-level representations of news, generating a knowledge-aware embedding vector using information of word embedding, entity embedding and embedding of the related contextual entities found in the KG combined in a multi-channel network. Collaborative Knowledge base Embedding [28] (CKE) combines a CF module with knowledge embedding, text embedding, and image embedding of items in a unified Bayesian framework.

Embedding-based methods show high flexibility in utilizing KG to assist recommender systems, but the adopted KGE algorithms in these methods are usually more suitable for in-graph applications such as link prediction than for recommendation, thus the learned entity embeddings in the KG are less intuitive and effective to characterize inter-item relations. Instead of directly plugging high-order relations into the model optimized for recommendation like path-based methods, these methods only encode them in an implicit manner [1].

2.2.5 Combination of both methods

There is a third set of models that combine both methods, addressing the limitations mentioned above. This thesis evaluates the performance of two of them:

2.2.5.1 RippleNet

RippleNet [1] is a method based on knowledge-graph-aware recommendation. The algorithm is designed for click-through rate (CTR) prediction, which takes a user-item pair as input and outputs the probability of the user engaging (e.g., clicking, browsing) the item.

The key idea behind RippleNet is preference propagation: for each user, RippleNet treats his historical interests as a seed set in the KG, then extends the user's interests iteratively along KG links to discover his hierarchical potential interests with respect to a candidate item. The concept of preference propagation is an analogy of actual ripples created by raindrops propagating on the water, in which multiple ripples superpose to form a resultant preference distribution of the user over the Knowledge Graph.

RippleNet addresses the limitations of embedding and path based methods by automatically discovering possible paths from an item in a user's history to a candidate item, without any sort of hand-crafted design as mentioned in section 2.2.3, and incorporating the knowledge graph embedding methods into recommendation naturally by preference propagation.

RippleNet was selected because it obtained the best results in rating based evaluation within state-of-the-art RS, measuring Accuracy and AUC, for several RS datasets.

2.2.5.2 Knowledge Graph Attention Network (KGAT)

Knowledge Graph Attention Network (KGAT) [4], is equipped with two designs to correspondingly address the challenges in high-order relation modeling. It uses recursive embedding propagation, which updates a node's embedding based on the embeddings of its neighbors, and recursively performs such embedding propagation to capture high-order connectivities in a linear time complexity. It incorporates an attention-based aggregation, which employs the neural attention mechanism to learn the weight of each neighbor during a propagation, such that the attention weights of cascaded propagations can reveal the importance of a high-order connectivity.

KGAT addresses the limitations of embedding and path based methods by avoiding the labor intensive process of manually crafting meta-paths, thus is more efficient and convenient to use, and factoring high-order relations into the predictive model, thus all related parameters are tailored for optimizing the recommendation objective.

KGAT was selected because it obtained the best results in ranking based evaluation within state-of-the-art RS, measuring recall@K and ndcg@K, for several RS datasets.

2.3 Available Resources for Knowledge Graphs

Nowadays, many organizations create their own Knowledge Graphs to organize their information into data and knowledge. Some examples of comercial KGs are Google's Knowledge Graph [1], Microsoft's Satori [2], Facebook's Entities Graph [3], etc.

Open knowledge graphs are published online, making their content accessible for the public good. The most prominent examples – DBpedia [37], Freebase [38], Wikidata [39], YAGO [40], etc. – cover many domains and are either extracted from Wikipedia, or built by communities of volunteers [3]. Prominent knowledge graphs, such as DBpedia and YAGO focus on extraction from info-box tables in Wikipedia.

The most recent version of the main DBpedia contains 4.8 million entities and 176 million triples, a triple being a set of <e1, rel, e2>, being $e1$ and $e2$ the entities related and *rel* the type of relationship between the entities. The ontology comprises 735 classes and 2,800 type of relations [33]. The latest release of YAGO, i.e., YAGO3, contains 4.6 million entities and 26 million triples. The schema comprises rouhgly 488,000 types of entities and 77 type of relations [33].

Freebase was aquired by Google and freezed in 2016, and the API has been shut down since [4], the avaliable version is a dump file that currently contains more than 50 million entities and 3 billion triples. Freebase's schema comprises roughly 27,000 entity types and 38,000 relation types. After the shutdown of Freebase, the data contained in Freebase was subsequently moved to Wikidata.

The Wikimedia Foundation proposed Wikidata as a centralised and collaboratively edited knowledge graph to supply Wikipedia – and arbitrary other clients – with data.

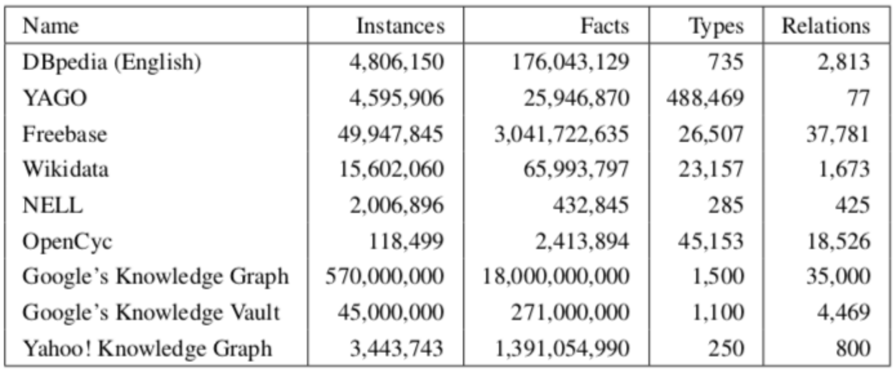
[1]https://developers.google.com/knowledge-graph
[2]https://searchengineland.com/bings-knowledge-repository-satori-just-got-a-lot-smarter-179800
[3]https://www.facebook.com/notes/facebook-engineering/under-the-hood-the-entities-graph/10151490531588920/
[4]https://developers.google.com/freebase

Under this vision, a fact could be added to Wikidata once, triggering the automatic update of potentially multitudinous articles in Wikipedia across different languages. To date, Wikidata contains roughly 16 million entities and 66 million triples. Its schema defines roughly 23,000 entity types and 1,600 types of relations.

Wikidata offers various access protocols and has received broad adoption, being used by Wikipedia to generate infoboxes in certain domains, being supported by Google, and having been used as a data source for prominent applications such as Apple's Siri amongst others [3].

Hence, the Wikidata API provides more structured information than DBPedia and YAGO, because it is not restricted to infoboxes, but acts as central storage for the structured data of the Wikimedia ecosystem [5].

The following figure summarizes the sizes of the most popular open source and commercial Knowledge Graphs:



Name	Instances	Facts	Types	Relations
DBpedia (English)	4,806,150	176,043,129	735	2,813
YAGO	4,595,906	25,946,870	488,469	77
Freebase	49,947,845	3,041,722,635	26,507	37,781
Wikidata	15,602,060	65,993,797	23,157	1,673
NELL	2,006,896	432,845	285	425
OpenCyc	118,499	2,413,894	45,153	18,526
Google's Knowledge Graph	570,000,000	18,000,000,000	1,500	35,000
Google's Knowledge Vault	45,000,000	271,000,000	1,100	4,469
Yahoo! Knowledge Graph	3,443,743	1,391,054,990	250	800

FIGURE 2.2: Sizes of popular Knoledge Graphs. Source: [33]

Microsoft Satori has not published official numbers of its KG metrics.

2.3.1 Wikidata

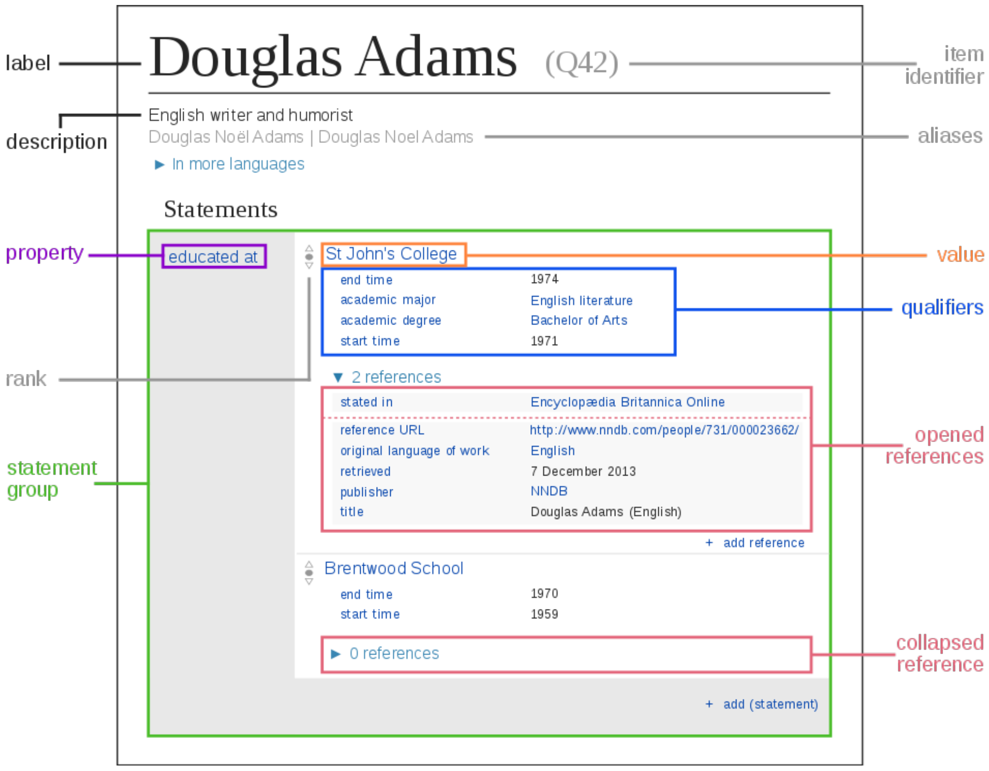
Wikidata [39] is a free and open knowledge base that can be read and edited by both humans and machines. The data in Wikidata is published under the Creative Commons Public Domain Dedication 1.0 [6], allowing the reuse of the data in many different scenarios. You can copy, modify, distribute and perform the data, even for commercial purposes, without asking for permission.

[5] https://meta.wikimedia.org/wiki/Wikidata/Notes/DBpedia_and_Wikidata

[6] https://www.wikidata.org/wiki/Wikidata:Licensing

Wikidata acts as central storage for the structured data of its Wikimedia sister projects including Wikipedia, Wikivoyage, Wiktionary, Wikisource, and others. Imposing a high degree of structured organization allows for easy reuse of data by Wikimedia projects and third parties, and enables computers to process it.

The Wikidata repository consists mainly of items, each one having a label, a description and any number of aliases. Items are uniquely identified by a Q followed by a number, such as Douglas Adams (Q42), as shown in the example below [7]:



Douglas Adams (Q42)

English writer and humorist

Douglas Noël Adams | Douglas Noel Adams

► In more languages

Statements

educated at

St John's College

end time	1974
academic major	English literature
academic degree	Bachelor of Arts
start time	1971

▼ 2 references

stated in	Encyclopædia Britannica Online
reference URL	http://www.nndb.com/people/731/000023662/
original language of work	English
retrieved	7 December 2013
publisher	NNDB
title	Douglas Adams (English)

+ add reference

Brentwood School

end time	1970
start time	1959

► 0 references

+ add (statement)

FIGURE 2.3: This diagram of a Wikidata item shows you the most important terms in Wikidata. Source: [8].

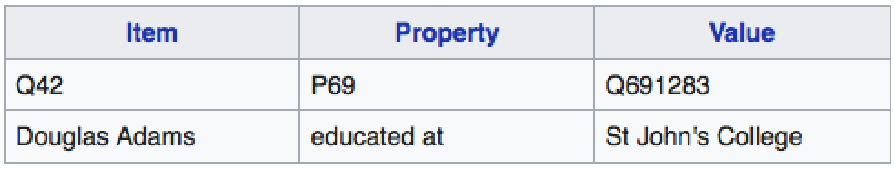
In Wikidata, concepts, places, people... are represented by items. Each item is accorded its own page. A statement is how the information known about an item is recorded in Wikidata. Statements describe detailed characteristics of an item and consist of a property and a value. Properties in Wikidata have a P followed by a number, such as *educated at (P69)*. The property in a statement describes the data value, and can be thought of as a category of data like *color* or *population*. The value in the statement is the actual piece of data that describes the item [9].

[7]https://www.wikidata.org/wiki/Help:Items
[9]https://www.wikidata.org/wiki/Help:Statements

Each property in Wikidata is assigned a pre-defined data type which restricts what can be added as its value. For example, only other items can be added as a value for the property *color*, only numbers can be added for the property *population*.

Statements are built by pairing a property with at least one value; this pair is at the heart of a statement. If an item by nature can have properties with multiple values (like children of a person or official languages of a country), it is perfectly acceptable to add each of these multiple values.

In order to obtain information about the place of education of Douglas Adams in Wikidata, first the page for Douglas Adams needs to be located *(Q42)*. The next step is to find the property *educated at* (P69) and obtain its related value, *St's Johns College*, (Q691283), as depicted in the figure below. This property could also have more related values associated, like the university he attended, and all of these values would be related to Adams.



Item	Property	Value
Q42	P69	Q691283
Douglas Adams	educated at	St John's College

FIGURE 2.4: Example of connection of Wikidata: Item, Property and Value.

For a person, a property could specify where they were educated, by specifying a value for a school. For buildings, geographic coordinates properties could be assigned by specifying longitude and latitude values. Properties can also link to external databases. A property that links an item to an external database, such as an authority control database used by libraries and archives, is called an identifier. Special Sitelinks connect an item to corresponding content on client wikis, such as Wikipedia, Wikibooks or Wikiquote.

Statements can also be expanded on, annotated, or contextualized with additional values, as well as optional qualifiers, references, and ranks. Statements also serve to connect items to each other, resulting in a linked data structure.

The following figure shows all of the properties and values related to the city of *San Francisco* (Q62). *San Francisco* is related to *California* by the property *in*, and also related to the value of its population by the property *population*.

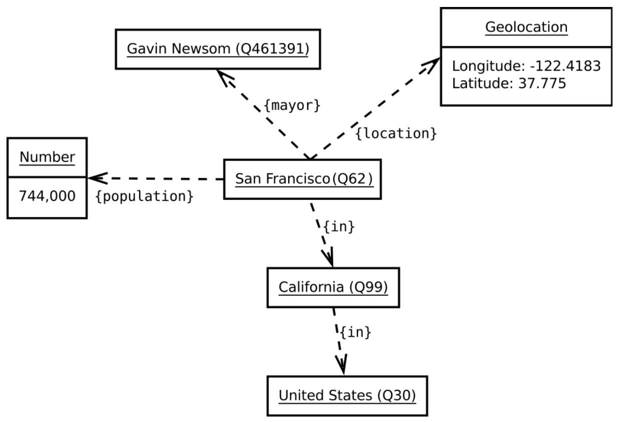


FIGURE 2.5: Interconnections of wikidata entities. Items and their data are interconnected. Source: [10]

The combination of all pages interconnected by properties constitutes the Knowledge Graph of Wikidata.

Chapter 3

Automatic Generation of Domain Knowledge Graph for Recommendation Systems

This chapter describes the proposed solution. The motivation behind the proposed system is building a flexible module to create domain adapted KGs from open source resources, that can be used in state-of-the-art recommendation algorithms. It is critical that the system can be adapted to different natures of reviews algorithms, such as movies, books, news..., and that the information retrieved in the KG is as complete as possible. The output of the system is also adapted to be used in any downstream hybrid Recommendation System.

The designed system will receive as an input a recommendation dataset (a dataset of item reviews such as movies, books...) , and will output a domain adapted Knowledge Graph, a directed graph with the item attributes and relations between the items of the dataset. For example, if the input is a dataset of movie reviews, the system will build a KG in which each movie will be linked to its director, cast, awards and genres, those movies sharing a director, cast member, award or genre will be also linked.

The Knowledge Graph is defined as a directed graph composed of subject-property-object triple facts. These triple facts will be extracted from external sources of information, in particular, the open source databases Wikipedia and Wikidata. The generated KG will be domain adapted because it will be composed of the relations shared between the items of the input dataset and the entities linked to those specific items.

The following figure shows a general scheme of the proposed system approach:

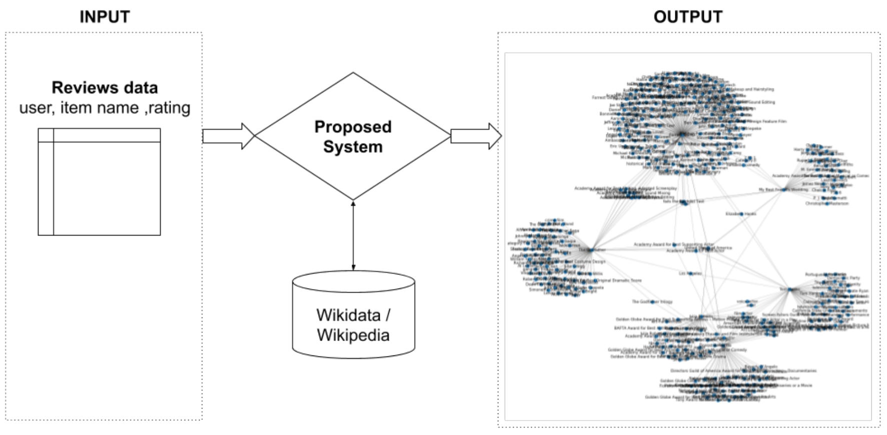


Figure 3.1: The designed system will receive as input a dataset of item reviews, and will return an output of a domain adapted Knowledge Graph.

The proposed system is composed of several modules. The developed system needs to be able to complete each of the following steps to create a KG using Wikidata for a given dataset:

1. **Data cleaning**: adaptation of the input review dataset
2. **Disambiaguation module**: Find the Wikidata id from item names in a reviews dataset for recommendation (eg. names of movies or names of books). First each item name has to be disambiguated, as an example, for the movie title "The Godfather", the system needs to find the Wikidata entry that corresponds to the movie instead of the book.
3. **Link extraction module**: Find the Wikidata entities liked to each Wikidata id item. Once the Wikidata entry is located, the system extracts all relations from the initial item to other Wikidata entries, for example the director, cast or any other listed properties, as explained in 2.3.

4. **Generation of domain adapted KG**: Create a final dataset with the KG shape. This output Knowledge Graph will then be used downstream as auxiliary data in two state-of-the-art Recommendation System, described in Section 2.2.5.

The process is iteratively repeated for the rest of the items (movies, books...) of the recommendation dataset.

The output Knowledge Graphs will be used in two Recommendation Systems, and the results will be evaluated in the following Chapter. For this purpose, the ids for each item in the reviews dataset and the Wikidata IDs in the KG must be aligned creating intermediate dictionaries. Then, the resulting files must be structured and adapted to the format expected by the Recommendation Systems. This process is detailed as follows:

- Adapt the generated dataset to the required shape by the Recommendation System (RippleNet and KGAT).
- Evaluate the results of the generated KG to be compared with the original papers.

The system general functioning scheme includes the proposed solutions for all the steps. They will be explained in detail trough the sections in this Chapter.

3.1 Designed System Architecture

The architecture of the modules will be detailed in the following sections and is depicted in the following figure:

3.1.1 Input data and cleaning

Datasets for recommendation systems are formed by reviews. Reviews are ratings that users give to the items presented to them. These ratings can be numerical (eg. a grade from 1 to 5), binary (eg. the item was clicked or not clicked) or ordinal (eg. a grade in terms of *good, medium or bad*). Reviews consist of four columns: *user id*, *rating*, *item id*

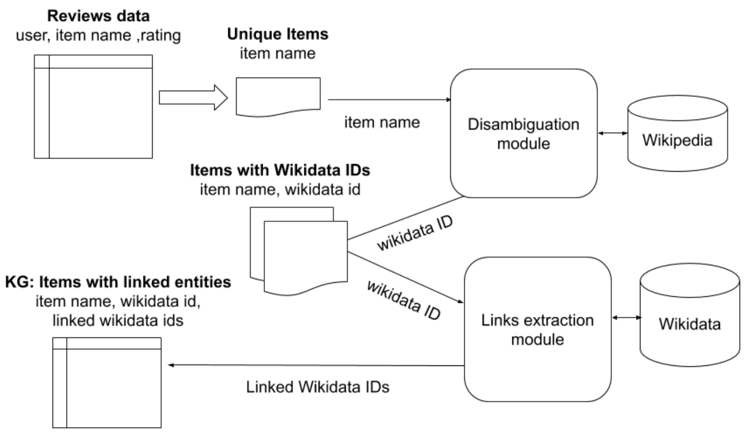
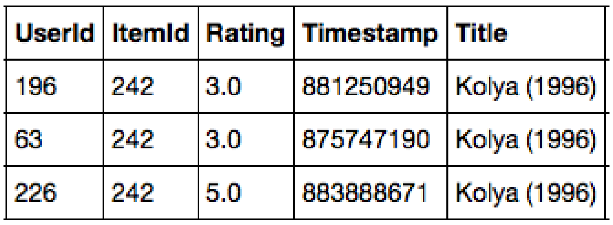


FIGURE 3.2: Wikidata ID and entities extraction module. The names of the items on the dataset are sent to the disambiguation module to infer their wikidata ID. The wikidata IDs per item are sent to the link extraction module to obtain the linked entities per item. This final set constitutes the generated domain adapted KG.



UserId	ItemId	Rating	Timestamp	Title
196	242	3.0	881250949	Kolya (1996)
63	242	3.0	875747190	Kolya (1996)
226	242	5.0	883888671	Kolya (1996)

FIGURE 3.3: Datasets for recommendation systems are formed by reviews. Composed of user id, item name and review .

and *item name.* The following figure is a sample of the movie recommendation dataset, Movielens:

Recommendation systems, as explained in Chapter 2, learn the preferences of users by analysing the ratings given to previously seen items.

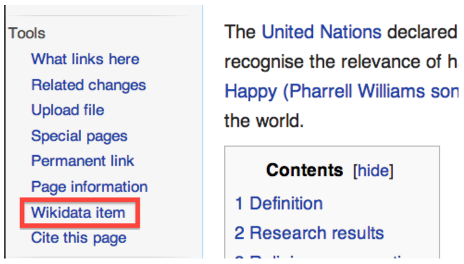
The proposed system will generate the domain adapted KG from the list of the names of items in the reviews dataset, hence the input review dataset is transformed into a list of unique strings representing the items' names. Additional words can be added to help the process of disambiguation, for example the word *movie* to differentiate movies with books with the same title. For the dataset presented above, the final list of items will contain: "*movie title*, *year of the movie*, movie", e.g. for the specificed example string representing the item would be *"Koyla, 1996, movie"*.

3.1.2 Disambiguation module

To generate the domain adapted Knowledge Graph from Wikidata, the first task is to find the Wikidata identifier corresponding to each item name in the recommendation dataset.

Wikidata items have an unique identifier or *QID* [1], as explained in Figure 2.3. In Wikidata, items are used to represent all the things in human knowledge, including topics, concepts, and objects. E.g.: the "1988 Summer Olympics", "love", "Elvis Presley", and "gorilla" are all items in Wikidata.

Finding a specific item name, such as the name of an specific movie, requires some method of disambiguation to match the item name with the specific item in Wikidata. As Wikidata is a part of the Wikimedia architecture, all Wikidata items have an entry as a Wikipedia page [2]. The following figure shows the link to the Wikidata entry from a Wikipedia page:



The United Nations declared
recognise the relevance of h
Happy (Pharrell Williams son
the world.

FIGURE 3.4: Link to Wikidata from Wikipedia Page.

So, finding an item in Wikipedia is equivalent to finding the entry in Wikidata. The benefit of using Wikipedia versus Wikidata to find specific items is that it contains the full content of the page, and that their text search engine is one of their core features.

The Wikipedia API [3] provides a *Wikipedia API full text search*, with the modules *query* and *search* that allow to search for page titles or content matching the string value sent, eg. *Koyla, 1996, movie*, [4] and retrieves the Wikipedia page with the best match.

Using this API is a very strong combination of all the powerful Wikipedia search engine [5] for the final objective of finding Wikidata QIDs. This search engine uses CirrusSearch

[1]https://www.wikidata.org/wiki/Help:Items
[2]https://en.wikipedia.org/wiki/Wikipedia:Finding_a_Wikidata_ID
[3]https://en.wikipedia.org/w/api.php
[4]https://en.wikipedia.org/w/api.php?action=help&modules=query%2Bsearch
[5]https://en.wikipedia.org/wiki/Help:Searching

[6], a MediaWiki extension that uses Elasticsearch to provide enhanced search features. All pages are stored in the wiki database, and all the words in the pages are stored in the search database, which is an index to practically the full text of the wiki. Each visible word is indexed to the list of pages where it is found, so a search for a word is as fast as looking up a single-record. Search term are matched against the content of the page to perform the search. Furthermore, for any changes in wording, the search index is updated within seconds. The API also allows to use search constraints with specific characters like exact phrase search, fuzzy search or name filters, like restricting the search to the page title (*intitle:*) or filtering to a specific page category (*incategory:*).

The resulting titles of the search are weighted by relevance, and heavily post-processed, 20 at a time, for the search results page. The response from this API query includes a list of the pages that match the full text search, in order of priority. The first page of the list will correspond to the best match to the full text search and the system will retrieve the Wikipedia *pageID*. The following figure represents the architecture of this method:

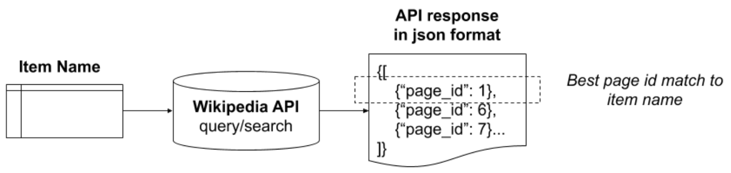
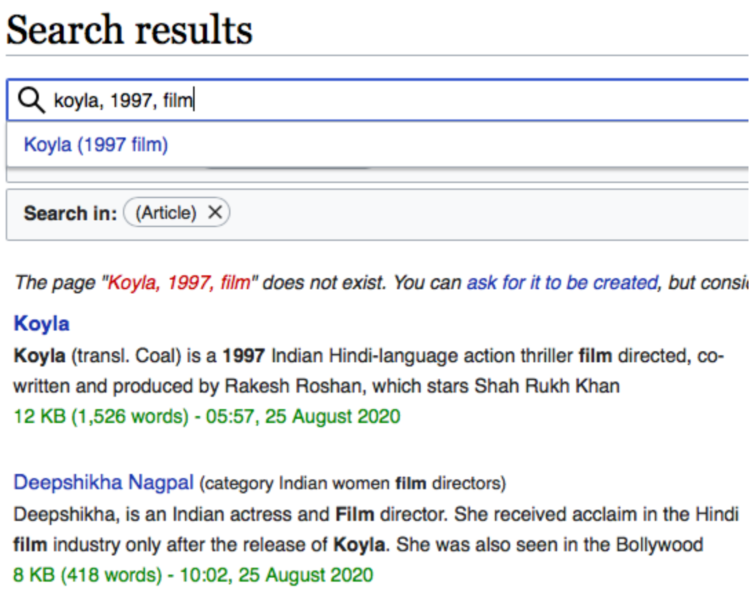

API response in json format
Item Name
Wikipedia API query/search
[{
{"page_id": 1},
{"page_id": 6},
{"page_id": 7}...
]}
Best page id match to item name

FIGURE 3.5: The Wikipedia API *query* and *search* allows to search for page titles or content matching the string value sent (eg. "book Little Women") .

The following figure is an example of the results retrieved for the example of using the API to search for the full text *Koyla, 1996, movie.* The first result retrieved from the query is the Wikipedia page for the specific movie:

The second method used in this module is the *Wikipedia API page properties retrieval.* This method complements the disambiguation, by finding the specific Wikidata QID of the disambiguated page. Once the text string of the item name has been disambiguated and the pageID has been retrieved, the module *pageprops* [7] allows to obtain the corresponding QID and other page properties if required [8]. This API function is

[6]https://www.mediawiki.org/wiki/Help:CirrusSearch
[7]https://en.wikipedia.org/w/api.php?action=help&modules=query%2Bpageprops
[8]https://www.mediawiki.org/wiki/API:Pageprops



Search results

koyla, 1997, film

Koyla (1997 film)

Search in: (Article)

The page "Koyla, 1997, film" does not exist. You can ask for it to be created, but consi

Koyla

Koyla (transl. Coal) is a **1997** Indian Hindi-language action thriller **film** directed, co-written and produced by Rakesh Roshan, which stars Shah Rukh Khan

12 KB (1,526 words) - 05:57, 25 August 2020

Deepshikha Nagpal (category Indian women **film** directors)

Deepshikha, is an Indian actress and **Film** director. She received acclaim in the Hindi **film** industry only after the release of **Koyla**. She was also seen in the Bollywood

8 KB (418 words) - 10:02, 25 August 2020

FIGURE 3.6: Wikipedia Full Text Search example

equivalent to clicking on the Wikidata item link shown in Figure 3.4, but the API allows for automatization of the task.

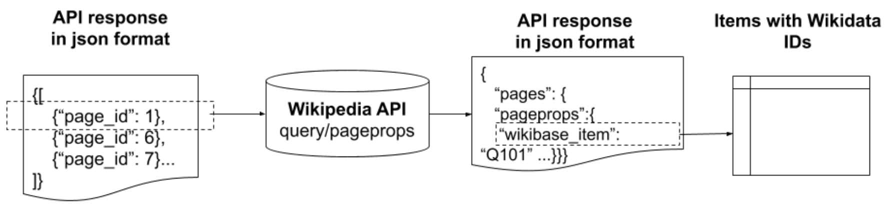
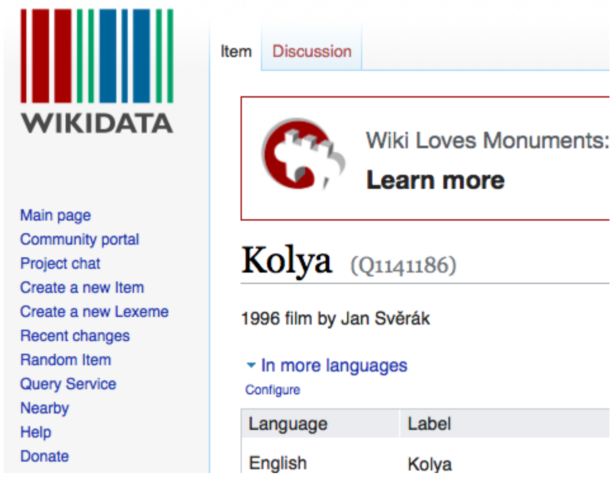


FIGURE 3.7: The Wikipedia API *query* and *pageprops* allows to search for Wikipedia Page IDs and obtain their respective Wikidata ID .

Obtaining this final QID as output allows the system to find the Wikidata entry corresponding to the input of the module, a text string of the item name. Continuing with the example, the retrieved Wikidata entry would be *Q1141186*, the Wikidata entry is the following:

3.1.3 Links extraction module

As mentioned in Section 2, Figure 2.3, each Wikidata item has associated Properties that have associated Values. Items can and should also be linked to each other. A link to another item is usually added as a ”Property”, and the name of the linked item is the ”Value”.

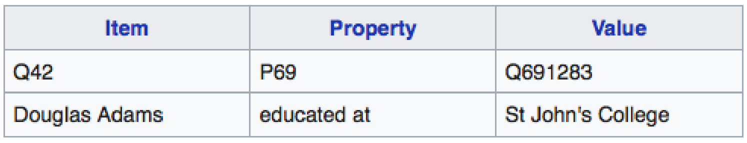


Kolya (Q1141186)

1996 film by Jan Svěrák

Language	Label
English	Kolya

FIGURE 3.8: Sample of Wikidata entry found after disambiguation.



Item	Property	Value
Q42	P69	Q691283
Douglas Adams	educated at	St John's College

FIGURE 3.9: Example of connection of Wikidata: Item, Property and Value.

Hence, after obtaining the Wikidata ID for each item name in the dataset, the next step is to use the Wikidata API to obtain the properties and values for each Wikidata ID.

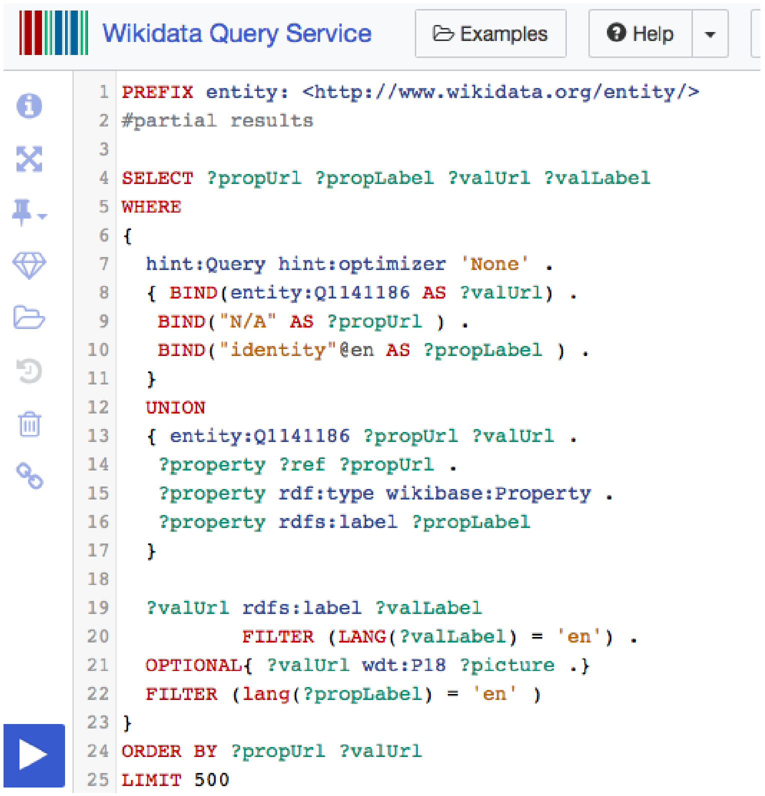
Wikidata provides an API to retrieve information from it knowledge base. The Wikidata Query Service (WDQS) [9] is a software package and public service designed to provide a SPARQL endpoint which allows to query against the Wikidata data set. SPARQL [10] is an RDF query language, that is, a semantic query language for databases.

Queries [11] retrieve logical combinations of triples. All information in Wikidata (and similar knowledge databases) is stored in the form of triples; when the query is run, the query service tries to fill in the variables with actual values so that the resulting triples appear in the knowledge database, and returns one or multiple results for each combination of variables it finds [12].

The query used by the system retrieves all properties for a given QID, with all the associated values for each properties. Figure 3.10 shows the query used for a specific

[9]https://www.mediawiki.org/wiki/Wikidata_Query_Service/User_Manual
[10]https://query.wikidata.org/sparqll
[11]https://www.wikidata.org/wiki/Wikidata:SPARQL_query_service/Wikidata_Query_Help
[12]https://www.wikidata.org/wiki/Wikidata:SPARQL_tutorial

example, the id *Q1141186* corresponding to the movie Kolya, in the Wikidata Query Service IDE:



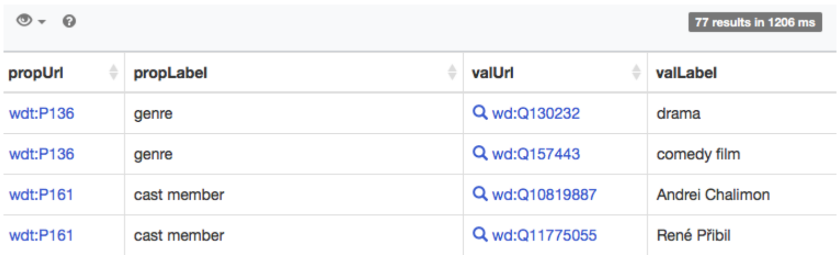
```
PREFIX entity: <http://www.wikidata.org/entity/>
#partial results

SELECT ?propUrl ?propLabel ?valUrl ?valLabel
WHERE
{
  hint:Query hint:optimizer 'None' .
  { BIND(entity:Q1141186 AS ?valUrl) .
   BIND("N/A" AS ?propUrl ) .
   BIND("identity"@en AS ?propLabel ) .
  }
  UNION
  { entity:Q1141186 ?propUrl ?valUrl .
   ?property ?ref ?propUrl .
   ?property rdf:type wikibase:Property .
   ?property rdfs:label ?propLabel
  }

  ?valUrl rdfs:label ?valLabel
          FILTER (LANG(?valLabel) = 'en') .
  OPTIONAL{ ?valUrl wdt:P18 ?picture .}
  FILTER (lang(?propLabel) = 'en' )
}
ORDER BY ?propUrl ?valUrl
LIMIT 500
```

FIGURE 3.10: SPARQL query to retrieve Wikidata triples

Retrieving all the results of the combinations allows the system to obtain all of the linked Values to a Wikidata ID, that is, all of the triples related to that item in the generated Knowledge Graph. In the example of Koyla, the API retrieved 77 triples, related to the genre, cast, director, producers, awards, country, subject of the film... . Figure 3.11 shows a sample of the triples.

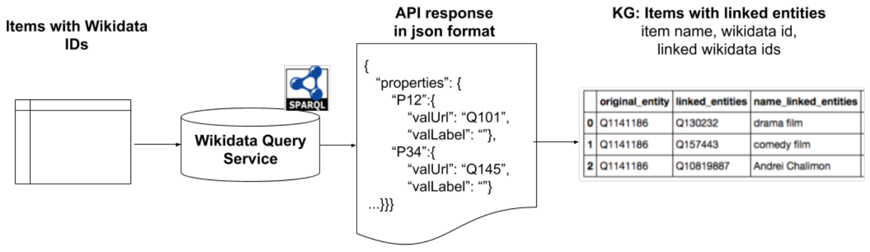


propUrl	propLabel	valUrl	valLabel
wdt:P136	genre	wd:Q130232	drama
wdt:P136	genre	wd:Q157443	comedy film
wdt:P161	cast member	wd:Q10819887	Andrei Chalimon
wdt:P161	cast member	wd:Q11775055	René Přibil

FIGURE 3.11: Query result to retrieve Wikidata triples

This process is automated in the system to use the SPARQL API and iteratively retrieve all properties and values for the corresponding QIDs of the items of the review datasets.

Hence, the system will take as input the list of Wikidata IDs generated by the disambiguation module, and will iterate trough the list making the corresponding SPARQL query for each of the items. The queries retrieve all properties and values liked to the item in Wikidata, that represent the corresponding triples in the final domain adapted KG The general architecture is shown in Figure 3.12.

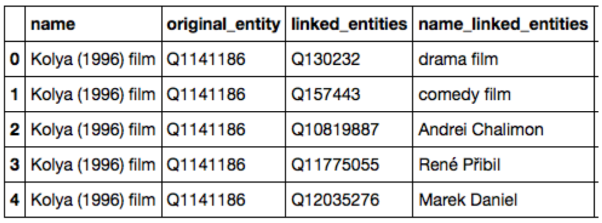

Items with Wikidata IDs
Wikidata Query Service
SPARQL
API response in json format
KG: Items with linked entities
item name, wikidata id, linked wikidata ids

```
{
  “properties”: {
    “P12”:{
       “valUrl”: “Q101”,
       “valLabel”: “”},
    “P34”:{
       “valUrl”: “Q145”,
       “valLabel”: “”}
 ...}}}
```

	original_entity	linked_entities	name_linked_entities
0	Q1141186	Q130232	drama film
1	Q1141186	Q157443	comedy film
2	Q1141186	Q10819887	Andrei Chalimon

Figure 3.12: A query to the SPARQL Wikidata endpoint allows to obtain all the properties and values linked to a Wikidata IS.

The resulting output will be a KG with the original item name (movie, book), the corresponding Wikidata ID and a row per linked entity found in Wikidata with their corresponding Wikidata ID and entity name:



	name	original_entity	linked_entities	name_linked_entities
0	Kolya (1996) film	Q1141186	Q130232	drama film
1	Kolya (1996) film	Q1141186	Q157443	comedy film
2	Kolya (1996) film	Q1141186	Q10819887	Andrei Chalimon
3	Kolya (1996) film	Q1141186	Q11775055	René Přibil
4	Kolya (1996) film	Q1141186	Q12035276	Marek Daniel

Figure 3.13: Samples of output Knowledge Graph for Movielens 1M .

3.1.4 Item alignment

Once the process is completed, the system will output a KG with the shape depicted in Figure 3.13. To be able to use this KG as a downstream input in a Recommendation System, the system must perform a matching and alignment process of the item ids.

The Recommendation Systems used in the project require the creation of intermediate dictionaries, linking the original ids of the reviews dataset and the Wikidata ids, and also the linked entities Wikidata ids to an internal id. This is done automatically by the system, since the schema of the generated Knowledge Graph is always the same. This module allows the output of the system to be flexible and consistent for different reviews dataset, and that this output can be used by different hybrid RS.

The output of this process will consist of two files. The first is called *item dictionary* that matches the recommendation dataset ids with the corresponding Wikidata IDs and an internal numerical id assigned to the item. The second is called *entity dictionary* and matches the linked entities found in the KG with an internal numerical ID used by the RS.

The following figure represents the files generated:

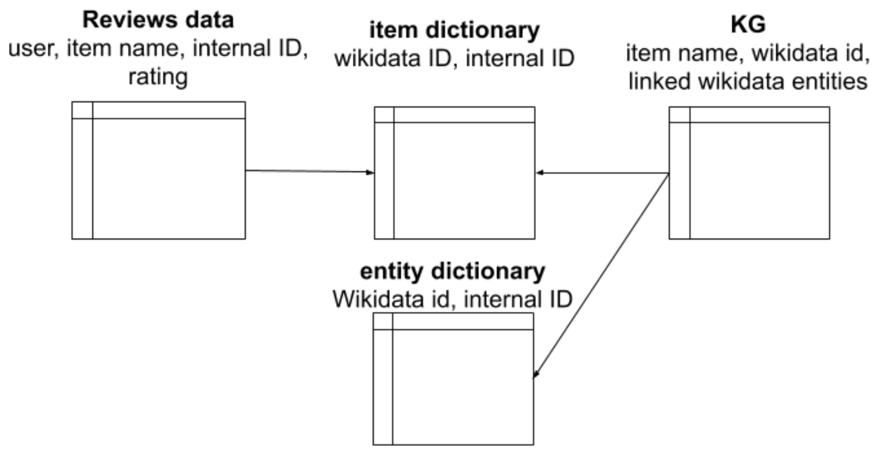


FIGURE 3.14: Alignment of IDs between reviews and generated KG. The ids for each item in the reviews dataset and the wikidata ids in the KG must be aligned creating intermediate dictionaries.

3.1.5 Integration with downstream RS and evaluation

The proposed system is a flexible architecture that allows to create domain adapted KGs from open source information, and the final goal is to use them in state-of-the-art recommendation algorithms. The architecture of the system allows to create domain adapted KGs for any reviews dataset, and adapts the output to be used in hybrid Recommendation systems. Hence, the architecture is reusable for different reviews dataset and RS algorithms. This has allowed to test the system with different reviews datasets and RS, as will be explained in the next Chapter.

The objective of this final module is to evaluate the generated KGs as external information in different hybrid RS. The final module will adapt and prepare the reviews datasets and generated KGs, split the data between train and test, and evaluate the results training a testing the state-of-the-art proposed algorithms. The architecture is depicted in Figure 3.15. This final module and architecture is also reused in different RS.

The steps covered by this final module are the following:

- The reviews data is split between train and test, the specific criteria to generate these files for each experiment is explained in 4.
- The train data, generated KG and intermediate dictionaries are used to train and build the Recommendation System.
- This built model is then used to predict recommendations for the test data, also using the KG and intermediate dictionaries.
- The predictions are evaluated against the real results in the test data to obtain the performance metrics.

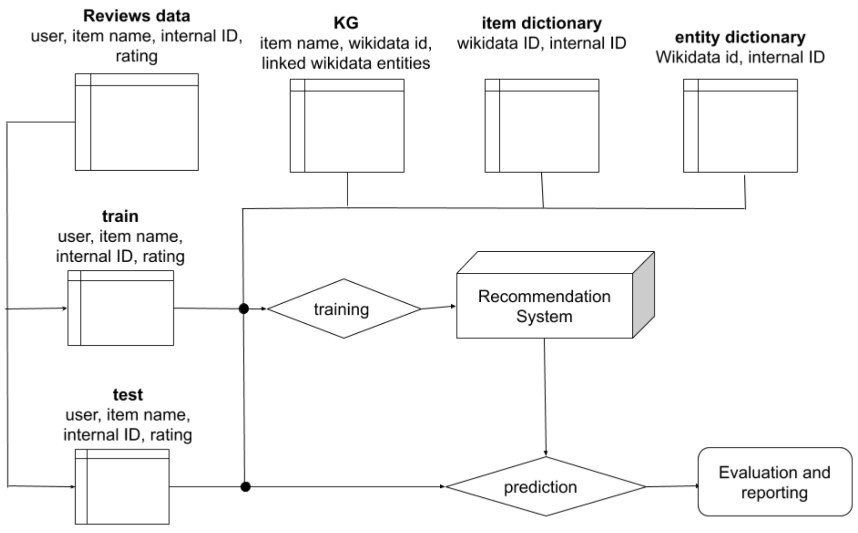


FIGURE 3.15: Recommendation System general scheme functioning. The reviews data is split between train and test, the train data, generated KG and intermediate dictionaries are used to train and build the Recommendation System. This built model is then used to predict recommendations for the test data, also using the KG and intermediate dictionaries. The predictions are evaluated against the real results in the test data to obtain the performance metrics .

In summary, the proposed system has been designed to be flexible to be adapted to different types of reviews datasets and create domain adapted KGs for each of them, and to be reused with different hybrid RS algorithms.

The final module can be easily adapted to multiple hybrid RS. This Thesis presents results for two state-of-the-art RS, and the only difference in the implementation was the structure of the files required by the RS, which was easily adapted. Hence, the system serves as a framework of evaluation of state-of-the-art hybrid RS.

Chapter 4

Evaluation and Discussion

The objective of this section is to evaluate the quality and performance of the Knowledge Graph generated by our system in a recommendation task. Two of the most recent and important state of the art recommendation algorithms using Knowledge Graphs are RippleNet [1] and KGAT [4]. One of the bottlenecks to benchmark and further improve or research these algorithms is that they used commercial KGs such as Microsoft Satori [1] or Freebase [38] (now Google Knowledge Graph [2]) as their KG. As Microsoft Satori and Google Knowledge Graph are not open source, it's not possible to use these solutions with new data in real applications without them. This section will evaluate our approach that aims to solve this problem.

We propose two types of Evaluation. The first evaluation is an Intrinsic Evaluation, where the objective is to evaluate the quality of the KGs generated by the proposed system against their commercial versions. For this purpose, we will measure the number of original items covered by the KG to measure coverage of the original dataset, the number of triples retrieved to measure the interconnectivity of the KG, and the number of entities retrieved outside the original items to measure the extension of the KG and the potential explainability that could be obtained from the KG. The second evaluation is an Extrinsic Evaluation, the objective is to measure the performance of the KGs in a downstream task, using the KGs as side information in a hybrid recommendation system.

[1] https://searchengineland.com/bings-knowledge-repository-satori-just-got-a-lot-smarter-179800
[2] [?]

We will measure the performance metrics related to rating and ranking evaluation, and compare them to the original results published in the papers.

The combined results of both evaluations show that our proposed system creates smaller KGs that are more domain adapted, measured in the intrinsic evaluation, and that our KGs have a similar efficiency in downstream tasks than commercial KGs, measured in the extrinsic evaluation. This could be specially relevant in systems that require faster computational times that can be achieved with smaller KGs, as real-time systems, or to save computational cost in high-scale systems, where the KGs generated by the proposed system could perform much better than the commercial counterparts.

4.1 Intrinsic Evaluation

The Intrinsic Evaluation will compare the output of the Knowledge Graph generated by the proposed solution against the KGs generated by commercial Knowledge Graphs: Microsoft Satori and Freebase for two separate Recommendation datasets: Movielens [5] and Amazon Book Review [6] [2], respectively. The KGs used in the papers are available as static datasets in the correspondent Github repositories [3] [4] of the algorithms, but the ids of the items, entities and relations are anonymized, so metrics will be computed on anonymized ids.

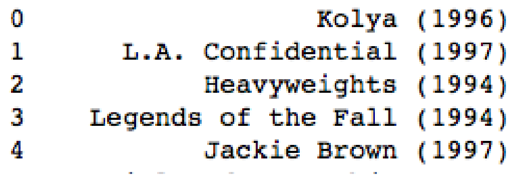
4.1.1 Evaluation Framework

4.1.1.1 Datasets

The proposed system was used to generate domain adapted Knowledge Graphs for the datasets Movielens 1M and Amazon Book Review, as was mentioned above. The input for the proposed system were each of the recommendation datasets, and the output each of the domain adapted KGs. The experimental setup for the two processes is described below:

[3] https://github.com/hwwang55/RippleNet
[4] https://github.com/xiangwang1223/knowledge_graph_attention_network

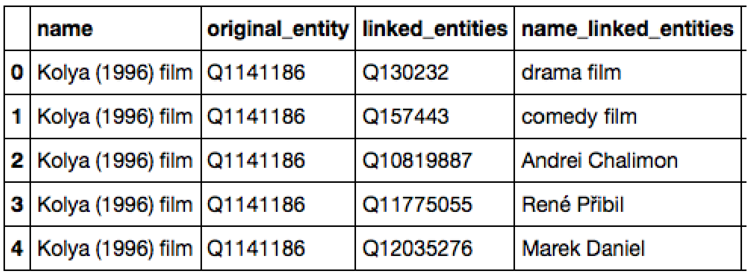
- Movielens [5]: is a dataset of movie reviews, where users rate movies with a value from 1 to 5. It contains 1M ratings of users on movies. The input dataset Movielens 1M has the following columns *'UserId', 'ItemId', 'Rating', 'Title' and 'Year'*. The input used in the proposed system was a string composed by the name of the movie, the year and the word *film*, as discussed in Section 3.1.2, to disambiguate movies from possible books with the same names, or other versions of the movie in other years. This method allowed the disambiguation of movie by restricting to films, excluding books, and the version of the film by including the year, shown in Figure 4.1:



```
0                  Kolya (1996)
1       L.A. Confidential (1997)
2            Heavyweights (1994)
3     Legends of the Fall (1994)
4            Jackie Brown (1997)
```

FIGURE 4.1: Samples of movie review entries from Movielens 1M.

The proposed system generates a KG that includes the items found in Wikidata, their correspondent ID, and the linked entities with their name and correspondent ID, shown in Figure 4.2:

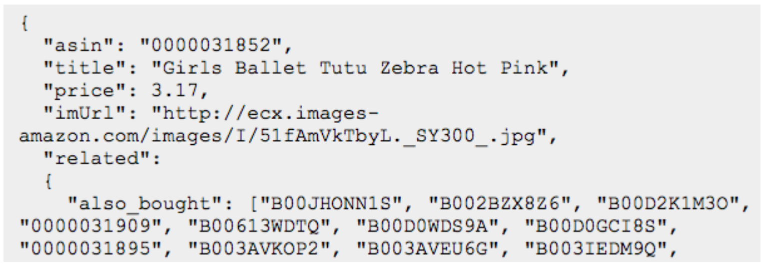


	name	original_entity	linked_entities	name_linked_entities
0	Kolya (1996) film	Q1141186	Q130232	drama film
1	Kolya (1996) film	Q1141186	Q157443	comedy film
2	Kolya (1996) film	Q1141186	Q10819887	Andrei Chalimon
3	Kolya (1996) film	Q1141186	Q11775055	René Přibil
4	Kolya (1996) film	Q1141186	Q12035276	Marek Daniel

FIGURE 4.2: Samples of output Knowledge Graph for Movielens 1M.

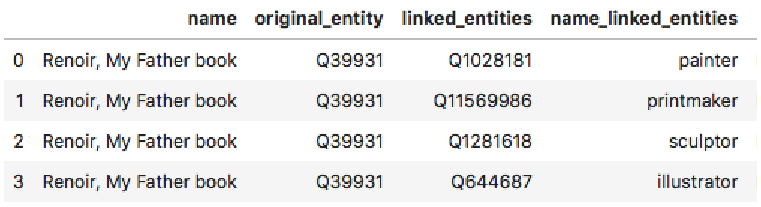
- Amazon Book Review [6]: The input data for Amazon Book Review is a *.json* file containing product reviews and metadata from Amazon. The metadata contains *'ASIN' (book unique identifier) and 'title'*. The input used in the proposed system was a string composed by the title of the book and the word *book*. This method allowed the disambiguation of books as shown in Figure 4.3:

  The proposed system generates a KG that includes the items found in Wikidata, their correspondent ID, and the linked entities with their name and correspondent ID, shown in Figure 4.4:



```
{
  "asin": "0000031852",
  "title": "Girls Ballet Tutu Zebra Hot Pink",
  "price": 3.17,
  "imUrl": "http://ecx.images-
amazon.com/images/I/51fAmVkTbyL._SY300_.jpg",
  "related":
  {
    "also_bought": ["B00JHONN1S", "B002BZX8Z6", "B00D2K1M3O",
"0000031909", "B00613WDTQ", "B00D0WDS9A", "B00D0GCI8S",
"0000031895", "B003AVKOP2", "B003AVEU6G", "B003IEDM9Q",
```

FIGURE 4.3: Samples of book review entries from Amazon Book Review.



	name	original_entity	linked_entities	name_linked_entities
0	Renoir, My Father book	Q39931	Q1028181	painter
1	Renoir, My Father book	Q39931	Q11569986	printmaker
2	Renoir, My Father book	Q39931	Q1281618	sculptor
3	Renoir, My Father book	Q39931	Q644687	illustrator

FIGURE 4.4: Samples of output Knowledge Graph Amazon Book Review

4.1.1.2 Knowledge Graphs

The output KGs will be evaluated against the following KGs commercial:

Movielens with Satori :

The experiments proposed in RippleNet [1] use Microsoft Satori [5] to build the Knoledge Graph for Movielens [5] datasets.

In the process followed by the authors of RippleNet to build the KG for MovieLens-1M from Microsoft Satory, the first process was to select a subset of triples from Satori whose relation name contains *movie* and the confidence level is greater than 0.9. Given the sub-KG, the IDs from Movielens were collected of all valid movies by matching their names with tail of triples (head, film.film.name, tail). For simplicity, items with no matched or multiple matched entities are excluded. Then the IDs are matched with the head and tail of all KG triples, all well-matched triples are selected from the sub-KG, and the process is extended to the set of entities iteratively up to four hops [1].

Amazon Book Review with Freebase :

The experiments proposed in the published paper of KGAT [2] use KB4Rec [41] to preprocess Amazon Book Review datasets and generate the KG, mapping items into

[5] https://searchengineland.com/library/bing/bing-satori

Freebase entities via title matching if there is a mapping available. KB4Rec is a linked dataset for research on knowledge-aware recommender systems. It aims to associate items from Recommendation Systems with entities from the available Freebase dump from 2013. KB4Rec organized the linkage results by linked ID pairs, which consists of a RS item ID and a KB entity ID. All the IDs are inner values from the original dataset. Amazon Book item ID are linked with a Freebase entity ID. Once such a linkage has been accomplished, KB4Rec allows researchers to reuse existing large-scale Knowledge Graph data for RS. For example, the book of from Amazon Book dataset has a corresponding entity entry in Freebase, the system is able to obtain its attribute information by reading out all its associated relation triples in KBs. Hence, with KB4Rec linkage and freebase dump, the sub-graph can be extracted. In particular, we consider the triplets that are directly related to the entities aligned with items, no matter which role (i.e., subject or object) it serves as. Distinct from existing knowledge-aware datasets that provide only one-hop entities of items, they also take the triplets that involve two-hop neighbour entities of items into consideration.

4.1.1.3 Metrics

Knowledge Graphs are made of triples that represent relations, for example *Madrid is the capital of Spain.* The formal definition of a triple is <e1, rel, e2>, being $e1$ and $e2$ the entities of the relation and rel the type of relation between the entities, in the example <Madrid, capital, Spain>.

The proposed system generates domain adapted Knowledge Graphs starting from Recommendation System datasets. These generated KGs will be evaluated against commercial KGs generated by Microsoft Satori and Freebase. It should be noted that the configurations of Freebase use a 2-hop entity extraction and Microsoft Satori proposes both a 1-hop and 2-hop extraction. Our proposed system uses a 1-hop entity extraction.

The reasoning behind the use of a 1-hop extraction is to generate domain adapted KG that will be smaller and faster to extract from the API, and that will still maintain the interconnectivity between the original items. Also, as stated in the paper by RippleNet: *A larger number of hops hardly improves performance but does incur heavier computational overhead according to experiment results. This is because any two items are probable to share a large amount of k-hop neighbours in the KG for a large k, even* *if there is no direct similarity between them in reality. The result motivates us to find a moderate hop number in RippleNet to explore users' potential interests as far as possible while avoiding introducing too much noise* [1]. By using a 1-hop configuration we force that our KG is more domain adapted, including less general terms that can link two items even if there is no real similarity between them, and the it is less computationally expensive.

The following metrics will be measured on the proposed Knowledge Graphs:

- **Number of original items in RS dataset**: items from the recommendation dataset found in the Knowledge Graph. The number of original items will measure how complete the domain adapted KG is compared to all the items in the recommendation dataset.
- **Number of triples**: number of relations found in the KG related to the original items. A triple is defined as <e1, rel, e2>. The number of triples will measure how big these KGs are.
- **Number of entities nor included in original RS dataset**: number of entities found in the triples (e2, e3...), excluding the original items. The number of entities, excluding the original items, will measure the diversity of the relations found and the complexity of the explicability that could be extracted from the set of triples.

4.1.2 Results

Dataset	KG	Number of original items	Number of triples	Number of entities
Movielens 1M	Proposed System (1 hop)	3,631	140,159	36,284
	Microsoft Satori (1 hop)	2,445	20,782	4,563
	Microsoft Satori (2 hops) *	2,445	178,049	55,583
Amazon Book Review	Proposed System (1 hop)	14,375	212,673	59,281
	Freebase (Google KG) (2 hop)	24,915	2,557,746	88,572

Table 4.1: Comparison of the domain adapted KGs generated by the proposed system against the commercial KGs used in the original papers .
* used in the paper by RippleNet

4.1.3 Discussion

In the Intrisic Evaluation, the KGs created by the proposed system were compared to their corresponding commercial KGs.

4.1.3.1 Movielens 1M

The proposed system was able to match more movies from the recommendation dataset to their corresponding entry than Microsoft Satori, hence the Number of original items is higher for the proposed system, showing that the disambiguation module of the system was able to find more accurately the entries for the corresponding entries in Wikidata than the process used to extract the entities in Satori. This could be due to the method used in the disambiguation module, whereas in their method to extract entities used exact title matching and *items with no matched or multiple matched entities are excluded*, our proposed system used best-matching of the entry using the full-text search of the combination of title and year of release of the movie. Hence items with multiple possible matches like movies with the same title but released in different years were disambiguated using the year. This could also be due with Wikidata having more items corresponding with movies, since it's a collaboratory open source of information continuously updated by the community.

In the 1-hop configuration, the proposed system retrieves a higher number of triples and related entities. This means that Wikidata contains more triples per original item in the dataset, meaning that it has more complete information per movie, linking to more entities.

Satori was able to retrieve more triples in total using the 2-hop configuration, and hence more entities in total. However, these triples and entities are not directly related to the original items, they have a distance of 2-hop. As mentioned before, with larger k-hop configurations, any two items are probable to share a large amount neighbours, but as the KG becomes more general and less domain adapted they could be interconnected by general terms, even if there is no direct similarity between them in reality.

4.1.3.2 Amazon Book Review

The proposed system was able to match less books from the recommendation dataset to their corresponding entry than Freebase using *KB4Rec*. This is due to Wikimedia not containing very old or not very popular books that are sold on Amazon, and could be fixed in the future as the content in Wikimedia is generated by the community, and the items in Wikidata are continuously updated by that content.

The difference in the number of triples is due to the fact that in the paper of KGAT, the generated KG using Freebase considers triplets that involve 2-hop neighbour entities of items into consideration, whereas our proposed system only considers 1-hop neighbour entities in its current architecture. However, as mentioned before, this means that our KG is more domain adapted, including less general terms that can link two items even if there is no direct similarity between them in reality.

Also, we can observe that even if the number of triples retrieved by Freebase with the 2-hop configuration is 12 times higher than the proposed system, the number of entities is only 1.5 times higher. This means that there are many more triple interconnections between the entities in Freebase, meaning that there are common concepts that serve as connections for many entities, which indicates that probably there are many general concepts that generate a big amount of triples to many entities.

4.1.3.3 Overall

It can be observed that in all the KGs, the number of entities is much smaller than the number of triples, that means that the KGs are very inter-connected, and there are common concepts that serve as connections for sets of groups of entities. However, this observation is especially relevant in the KG of Amazon Book Review with Freebase.

The main differences found between the KGs generated by the proposed system and the commercial KGs are the fact that the proposed system uses a 1-hop configuration, whereas the commercial KGs use a 2-hop configuration, therefore they contain more triples and more final entities. However, as mentioned before, this means that our KG is more domain adapted, including less general terms that can link two items even if there is no direct similarity between them in reality. The next Section will evaluate the KGs extrinsically, and we will analyse the effect of this difference in hops on the performance of state-of-the-art hybrid RS.

4.2 Extrinsic Evaluation

The Extrinsic Evaluation will benchmark the performance of the algorithms RippleNet and KGAT using Knowledge Graph generated by the proposed solution and compare the results with the original results in the published papers using Microsoft Satori and Freebase.

The results of both RS with the proposed system use the same hyper parameters and configuration as the default shown in the papers for the specific datasets.

4.2.1 Evaluation Framework

The knowledge-graph-aware recommendation problem is formulated as follows.

In a typical recommender system, let $\mathcal{U} = \{u1, u2, ...\}$ and $\mathcal{V} = \{v1, v2, ...\}$ denote the sets of users and items, respectively. The user-item interaction matrix $Y = \{y_{uv} | u \in \mathcal{U}, v \in \mathcal{V}\}$ is defined according to users' implicit feedback, where:

$$y = \begin{cases} 1, & \text{if interaction (u, v) is observed} \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

A value of 1 for y_{uv} indicates there is an implicit interaction between user u and item v, such as behaviors of watching, reading, clicking, browsing, etc. In addition to the interaction matrix Y, we also have a knowledge graph $\mathcal{G}$ available, which consists of massive <e1, rel, e2> triples.

The proposed algorithms take a user u and an item v as input, and outputs the predicted probability that user u will watch, read or click item v.

The code for both algorithms was adapted from their respective Github repositories, to use our proposed KG as input side information:

- RippleNet: `https://github.com/hwwang55/RippleNet`
- KGAT:`https://github.com/xiangwang1223/knowledge_graph_attention_network`

4.2.1.1 Dataset preparation

MovieLens-1M reviews are numerical values, meaning that the users rate the movies with a mark from 1 to 5 being 1 the lowest and 5 the highest, this is considered explicit feedback data. This rating is transformed into implicit feedback where each entry is marked with 1 indicating that the user has rated the item with a mark of 4 or 5. For each positive instance, the dataset contained a random movie that the user had not watched, marked as 0.

For Amazon Book Review, for each user the books that were reviewed were treated as a positive instances, marked as 1. For each positive instance in the train data, the train data included a negative instance, marked as 0, of a random book that the user did not consume before. For each user in the test set, all the items that the user has not interacted with are treated as the negative items.

4.2.1.2 Algorithms

RippleNet :

RippleNet is an end-to-end framework that naturally incorporates the knowledge graph into recommender systems. Similar to actual ripples propagating on the water, RippleNet stimulates the propagation of user preferences over the set of knowledge entities by automatically and iteratively extending a user's potential interests along links in the knowledge graph. The multiple "ripples" activated by a user's historically clicked items are thus superposed to form the preference distribution of the user with respect to a candidate item, which could be used for predicting the final clicking probability [1].

For each dataset, the ratio of training is 60% of the data, evaluation set is 20% of the data, and test set is 20% of the data.

KGAT :

Knowledge Graph Attention Network (KGAT) explicitly models the high-order connectivities in KG in an end-to-end fashion. It recursively propagates the embeddings from a node's neighbours (which can be users, items, or attributes) to refine the node's embedding, and employs an attention mechanism to discriminate the importance of the neighbours. KGAT is conceptually advantageous to existing KG-based recommendation methods, which either exploit high-order relations by extracting paths or implicitly modelling them with regularization [2].

To ensure the quality of the dataset the data pre-processing uses 10-core setting, i.e., retaining users and items with at least ten interactions.

For each dataset, the data pre-processing select 80% of interaction history of each user to constitute the training set, and treat the remaining as the test set. From the training set, 10% of interactions are selected as validation set to tune the hyper-parameters.

Since the domain adapted KG created by the proposed system was smaller than the KG proposed in the paper, some of the users did not match the 10-core setting mentioned, hence the user set and the train and test set were reduced:

Dataset	KG	Number users	Number items	Number reviews train	Number reviews test
Amazon Book Review	Proposed System	68.605	14.375	269.024	58.377
	Freebase	70.679	24.915	269.024	71.749

Table 4.2: Number of users and items comparison for Amazon Book Review.

4.2.1.3 Metrics

Metrics in Recommendation System can be measured from two perspectives: rating-based classification and ranking-based user models. For *rating-based* the probabilities of the output of the model are treated as a binary, measuring the performance as simulating a Click-Through-Rate (CTR) prediction, meaning a user clicked or did not click an item presented to them. For *ranking-based* user models, the models outputs the top-K items with the highest predicted click probability, and metrics are measured on that set of items. The papers studies used different approaches to evaluation, explained below:

		True value	
		Positive	Negative
Prediction	Positive	TP	FP
	Negative	FN	TN

Table 4.3: Confusion Matrix for Prediction and associated abbreviations.

RippleNet Used a rating based evaluation, measuring Accuracy and AUC to evaluate the performance of CTR prediction.

- AUC: A ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate (TPR) is a synonym for recall and False Positive Rate (FPR). Plotting these two rates in axis shows how much model is capable of distinguishing between classes. AUC stands for "Area under the ROC Curve". Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. That is, AUC measures the entire two-dimensional area underneath the entire ROC curve [6]

[6] urlhttps://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

- Accuracy: measures the rate or correct predictions:

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{4.2}$$

KGAT Used a ranking based evaluation. The model outputs the user's preference scores over all the items, except those that were already positive for each user in the the training set, to avoid double counting. These preference score for the items are ranked for each user, building a rank from most-preferred to least-preferred items. The top-K recommendation represent the K most-preferred items for each user predicted by the system. These top-K recommendation are compared to the *gold standard*, the real preferences of the user shown in the test set.

To evaluate the effectiveness of top-K recommendation and preference ranking, the paper adopted two widely-used evaluation protocols: recall@K and ndcg@K. By default, K is set to 20. The paper reported the average metrics for all users in the test set.

- Precision@K, being K the items with higher rating given by the user. Measures how many of the top-K items with higher probability were actually correct:

$$precision = \frac{TP_{atKtop}}{K_{higher_rating}} \tag{4.3}$$

- Recall@K, being K the items with higher rating given by the user. Measures how many of the items predicted positive by the model are in the top-K and are correct:

$$recall = \frac{TP_{atKtop}}{TP + FP} \tag{4.4}$$

- NCGD@K: Normalized Discounted Cumulative Gain, is a metric that is commonly used in information retrieval and measures the quality of a ranking, being K the items with higher probability predicted by the system [7]:

$$DCG = \sum_{i=1}^{K_{higher_probability}} \frac{2^{relevance(i)-1}}{\log(i+1)} \tag{4.5}$$

[7] https://mlexplained.com/2019/05/27/learning-to-rank-explained-with-code/

Where $relevance(r)$ is the relevance score of each item consumed, where the higher an item is ranked, the higher the value of the relevance.

$$NDCG = \frac{DCG}{\max DCG} \tag{4.6}$$

and $\max DCG$ is the DCG you would get with a perfect ranking.

4.2.2 Results

RippleNet The following table shows the results of the corresponding metrics for RippleNet:

Dataset	KG	AUC	Accuracy
Movielens 1M	Proposed System	92.2 %	84.9 %
	Microsoft Satori (*)	92.1 %	84.4 %

Table 4.4: Results for RippleNet with Movielens 1M
(*) obtained from original paper, using 2-hops [8]

KGAT The following table shows the results of the corresponding metrics for KGAT:

Dataset	KG	Recall @20	NCGD @20	Precision @20
Amazon Book Review	Proposed System	11.9 %	9.548 %	1.76 % %
	Freebase (*)	14,89%	10,06%	1,5% %

Table 4.5: Results for KGAT with Amazon Book Review
(*) obtained from original paper

4.2.3 Discussion

The results in the Extrinsic Evaluation measure the performance of two Recommendation system using the domain adapted Knowledge Graphs generated by the proposed system against the performance using commercial KGs.

4.2.3.1 RippleNet

The performance obtained for both AUC and accuracy using the KG generated by proposed system is almost equal, even a bit higher than the performance obtained using the commercial KG Microsoft Satori.

This shows that even if the Knowledge Graph generated by the system is smaller, as discussed in the section above, the algorithm is able to learn form the 1-hop relationships between the movies and perform at a high level. The amount of triples and entities in the KG generated by the proposed system has many more triples and related entities than the 1-hop version of the commercial KG of Microsoft Satori. This means that there are many more direct relationships to the original items and that the direct information to them is richer, meaning that our open source proposed solution is able to generate more complete information than the commercial solution about the original items.

The version of the KG by Microsoft Satori used in the paper was the 2-hop version. This means that even if the KG is slightly bigger and includes more entities, these triples and entities are at a 2-hop distance of the original items, and can include many general concepts that do not represent a real relationship between the items. Our proposed domain adapted KG is able to provide richer information that directly links to the original items using the 1-hop restriction, hence more relevant information, even if the number of triples and entities is a bit smaller.

4.2.3.2 KGAT

The performance obtained for Precision and Hit at K = 20 is higher using the KG generated by proposed system, whereas the results for Recall is higher using the commercial KG Freebase. The performance for NCGD is a bit lower for the proposed system, but it still above the benchmark of state-of-the-art recommendation systems shown in the paper [2].

This shows that using a smaller domain adapted KG, with a 1-hop configuration can benefit some of the metrics.

- The precision being higher means that the algorithm was better at giving a high probability to those items that were really consumed, meaning that in the top-20 items recommended, more of those were really consumed.
- The recall being lower means that the ratio of all the items the predicted as positive versus how many of them were in the top K was smaller.
- As the value of NCGD is similar, the rank or order of the items showed in the top 20 is still relevant.

Taking all of these perspectives into consideration, we can conclude that the inclusion of the proposed KG in the algorithm produces a higher rate of true positives, hence there are more positives in the top 20, and also a higher rate of positives in the general base, reducing the recall rate in the top 20.

Hence, using the KG generated by the proposed system, although its size is smaller than the commercial KG, can achieve better results at some of the metrics. There is a tradeoff between recall and precision as discussed above that should be evaluated by the application of the model.

Chapter 5

Conclusions and Future Work

This Thesis presents a system to generate domain adapted Knowledge Graphs using open source information. These Knowledge Graphs can be used in hybrid Recommendation Systems, that use linked knowledge on the items as side information, combining both Collaborative Filtering and Content Base Filtering strategies. The results show that the proposed system is able to create domain adapted Knowledge Graphs from open source information for recommendation datasets, and that the KGs generated are able to compete with their commercial versions. These KGs can be used as auxiliary information in hybrid Recommendation Systems, achieving performance metrics that are similar to the metrics obtained with commercial KGs.

Knowledge Graphs are used in many fields, recently they became more popular in a branch of research on Recommendation System that has focused on incorporating side information to address problems such as sparsity and cold-start of items and users. One of the proposed solutions to this problem is to use a domain adapted KG, a graph that maps the relationships between items of the recommendation dataset. The State-of-the-Art papers addressing this research question evaluate Recommendation Systems that use commercial KGs such as Microsoft Satori or Freebase (owned by Google). The ability to generate open source domain adapted Knowledge Graphs is critical in the development of the field of Recommendation Systems with hybrid approaches, without having to depend on commercial information owned by specific companies.

These are the reasons why this thesis is a step further to make science accessible and replicable for all the scientific community, empowering the development of new knowledge areas.

Our system has been tested generating domain adapted KGs for two Recommendation System datasets, Movielens 1M and Amazon Book Review. The quality of the generated KGs has been tested intrinsically, comparing them to commercial KGs: Microsoft Satori and Freebase (owned by Google). The application of the KGs has been tested extrinsically, using them as external information in two KG-based hybrid Recommendation Systems: RippleNet and KGAT. The metrics used measured both a rating based approach and a ranking based approach of the generated results.

The results show that the proposed system is able to match more items from Movielens 1M than Microsoft Satori but less items from Amazon Book Review than Freebase. The size of the KGs generated is also smaller in terms of triples and linked entities retrieved, since the proposed system extracts the 1-hop connections to the original entities, whereas the KGs generated by Satori ans Freebase both contain 2-hop connections. However, in terms of performance of the generated KGs in the proposed Recommendation Systems, our KGs were able to match the performance of the commercial KGs. In a rating based approach the proposed system overperform in precision@20 but underperformed in recall@20, this analysis showed that using the proposed system produced that more items were predicted as positive. This tradeoff, however, is not bad per se, as some of the applications may require a higher precision and do not care that much about the recall.

Even if we can consider this Thesis as a proof of concept, our work leaves other paths open for exploration. We have showed that it is possible to build domain adapted Knowledge Graphs using open source information in Wikidata. However, the extraction of the KGs extracted was restricted to 1-hop, hence the size of the KGs was much smaller compared to their commercial versions. We have shown that our proposed system creates smaller KGs that are more domain adapted, that have a similar efficiency in downstream tasks than commercial KGs. This could be specially relevant in systems that require faster computational times that can be achieved with smaller KGs, as real-time systems, or to save computational cost in high-scale systems. Thinking about future work in this area, the system can be configured to also extract 2-hop relationships. The size of the KG has been proven not to be a critical factor, hence the tradeoff between the size of the KG with resources to generate it and the performance of the model can be evaluated.

Another aspect that can be improved in our system is to extract the type of relationship in each of the triples, using the *Property* field in Wikidata, something that the commercial KGs include, and could be useful for future work in explainability of the generated recommendations.

In terms of Evaluation, this Thesis has used ranking and rating based metrics, reproducing those that were measured in the original papers. However, there are problems related to restricting the evaluation of the metrics to these two types. One of the problems is related to the diversity of the results, a system recurrently recommending only very similar items to the user can create *echo chambers*, only suggesting the same type of content over and over again. Another area to consider is the ability to use the structure of the KGs to explain the recommendations of the system, such as *watch Forrest Gump because you liked this other movie by Tom Hanks.* This information in terms of linked path from item to item can be extracted from KGs. Hence, for future work there are more metrics that can also be used to evaluate the performance of the proposed KGs, such as diversity of the items recommended or the use of the KG to explain the recommendations.

Appendix A

Regulatory Framework

The software developed by the author for this Thesis is available, including the main system and the experiments performed. The code is licensed under the MIT license [1]. It can be re-licensed under other licenses and permits reuse within proprietary software, provided that either all copies of the licensed software include a copy of the MIT License terms and the copyright notice, or the software is re-licensed to remove this requirement.

The version of Python [42] used in the software is 3. It is declared Open Source and GPL compatible. "Open source software is software that can be freely used, changed, and shared (in modified or unmodified form) by anyone." [42]. The platform to execute the code using GPUs was Google Collab, with is free to use with a Google Account [2].

In Wikidata, all structured data from the main and property namespace is available under the *Creative Commons CC0 License*, with no rights reserved [3].

The dataset used, Movielens may be used for any research purposes, researchers must acknowledge the use of the data set in publications and may not use this information for any commercial or revenue-bearing purposes without first obtaining permission from the University of Minessota [4]. Amazon Product Review mentions that they do not own the

[1] `https://opensource.org/licenses/MIT`
[2] `https://research.google.com/colaboratory/faq.html`
[3] `https://www.wikidata.org/wiki/Wikidata:Copyright`
[4] `http://files.grouplens.org/datasets/movielens/ml-25m-README.html`

data, and as such we are not in a position to offer a license or control its use, however we request that the data be used for research purposes [5].

RippleNet and KGAT use an MIT license [6] [7]

Other Python libraries used in the software are:

Numpy [43]: simplified or new-license BSD [44]. They publish it explicitly in their webpage [45] **Pandas** [46]: simplified or new-license BSD [44]. Can be found at [47]. **Tensorflow**: [48]: Apache License 2.0 [49].

Since I could not have done any of these work without the Open Source tools I have used, I think that it is only fair publishing the code for the application so others can reuse it. In my opinion, research can only move forward if we share our applications and discoveries with others and keep ”standing on the shoulders of giants”[8].

[5]`http://deepyeti.ucsd.edu/jianmo/amazon/index.html`
[6]`https://github.com/hwwang55/RippleNet/blob/master/LICENSE`
[7]`https://github.com/xiangwang1223/knowledge_graph_attention_network/blob/master/LICENSE`
[8]Phrase known as an expression of Isaac Newton in a letter to Robert Hook in 1676

Appendix B

Project Planning and Budget

B.1 Project Planning

The idea of working on this Thesis started in June of 2019, when discussing possible thesis projects with Miguel Gonzalez-Fierro, Data Scientist at Microsoft. Microsoft is working on an Open Source Repository on Recommendation Systems [1] [50]. One of the issues they were facing was that some of the results on State-of-the-Art papers, such as RippleNet [1] were not reproducible on new data as the KGs they use was Microsoft Satori. They were looking for open source alternatives to Satori to create reproducible experiments with the algorithm.

After talking to Jorge Carrillo, my supervisor, we agreed that creating a system that coudl extract domain adapted KGs from Open Source information, and to evaluate their performance against commercial KGs could be a suitable topic for the Thesis.

The first phase of the project, ranging from September to November, included the research on Open Source Knowledge Graph repositories that could serve was the source of information for the proposed system. After choosing Wikidata, I developed the system to extract the domaing adapted KG from a list of items. The resulting system was evaluated by Data Scientist at Microsoft and published on their repository.

The second phase consisted on replicating the results of RippleNet using the proposed system. The first step was to verify that the results from the paper could be replicated,

[1] https://github.com/microsoft/recommenders

using the code and data in the algorithm's repository. As the algorithm required a GPU, and I did not have one, I used the free instances of GPU provided by Google Collab. After verifying the original results could be reproduced, I proceeded to do the experiments with the KG generated by the proposed system. First, I created a domain adapted KG based on the dataset Movielens 1M using the developed system. Then, I adapted the files to the format expected by RippleNet. Then, I reproduced the results using Google Collab. The resulting experiments was evaluated by Data Scientist at Microsoft and published on their repository.

For the third phase, Jorge suggested that we should evaluate the proposed system against another commercial Knowledge Graph. We chose to replicate the results of another state of the art algorithm, KGAT, that used Freebase (Google's Knowledge Graph). The process followed was similar to the process mentioned above, first replicating the results of the paper using the code and data in the algorithm's repository, then using the proposed system to create a domain adapted KG for Amazon Book Review [] and then reproducing the results of the paper with the new KG using Google Collab. The exception in this case was that the data preparation phase was harder, since the recommendation dataset was not a ready-to-use csv file, the reviews were stored in *.json* files, and I had to match the ASIN code of the book to the book title.

The fourth phase consisted on finishing the evaluation and writing the content of the Thesis.

The approximate time dedicated to this Thesis was 975 hours. The hours are calculated from September of 2019.

- 1st phase: September - November (90 days) average of 3 hours daily working on the project
- 2nd phase: December - February (90 days) average of 3 hours daily
- 3rd phase: March - July (150 days) average of 2 hours daily
- 4th phase: July - September (75 working days - 15 vacation) average of 3 hours daily

There were meetings appointed with my supervisor every other week.

The following budget is based on base salaries for a Senior Engineer in Spain, pre taxes, and the hours worked in this Thesis. It was assumed that the working hours in a month are 240, 8 hours per day.

Description	Units	Amount	Unitary price	Total
Junior Engineer	h	975	25 €	24,375 €
MacBook Air 13"	uds.	1	1,229 €	1,229 €
TOTAL PROJECTED COST				$\boxed{25{,}604 \text{ €}}$

TABLE B.1: Project Budget Estimation

Bibliography

[1] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 417–426, 2018.

[2] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. KGAT: knowledge graph attention network for recommendation. In *KDD*, pages 950–958, 2019.

[3] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs, 2020.

[4] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 950–958, 2019.

[5] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[6] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

[7] Wayne Xin Zhao, Gaole He, Kunlin Yang, Hongjian Dou, Jin Huang, Siqi Ouyang, and Ji-Rong Wen. Kb4rec: A data set for linking knowledge bases with recommender systems. *Data Intelligence*, 1(2):121–136, 2019. doi: 10.1162/dint_a_00008. URL https://doi.org/10.1162/dint_a_00008.

[8] Poonam B Thorat, RM Goudar, and Sunita Barve. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications*, 110(4):31–36, 2015.

[9] Elaine Rich. User modeling via stereotypes. *Cognitive science*, 3(4):329–354, 1979.

[10] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[11] Jesus Bobadilla, Fernando Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 07 2013. doi: 10.1016/j.knosys.2013.03.012.

[12] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201, 1995.

[13] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[14] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[15] Nicholas J Belkin and W Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[16] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[17] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pages 151–161, 2019.

[18] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[19] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35 (12):61–70, 1992.

[20] Peter Brusilovski, Alfred Kobsa, and Wolfgang Nejdl. *The adaptive web: methods and strategies of web personalization*, volume 4321. Springer Science & Business Media, 2007.

[21] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, 1995.

[22] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. A survey of serendipity in recommender systems. *Knowledge-Based Systems*, 111:180–192, 2016.

[23] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[24] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172, 2013.

[25] Amit Sharma and Dan Cosley. Do social explanations work? studying and modeling the effects of social explanations in recommender systems. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1133–1144, 2013.

[26] Rose Catherine and William Cohen. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 325–332, 2016.

[27] Xu Chen, Hanxiong Chen, Hongteng Xu, Yongfeng Zhang, Yixin Cao, Zheng Qin, and Hongyuan Zha. Personalized fashion recommendation with visual explanations

based on multimodal attention network: Towards visually explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 765–774, 2019.

[28] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362, 2016.

[29] Piero Andrea Bonatti, Stefan Decker, Axel Polleres, and Valentina Presutti. Knowledge graphs: New directions for knowledge representation on the semantic web (dagstuhl seminar 18371). In *Dagstuhl Reports*, volume 8. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[30] Richard H Richens. Preprogramming for mechanical translation.

[31] Fritz Lehmann. *Semantic networks in artificial intelligence*. Elsevier Science Inc., 1992.

[32] Sri Nurdiati and Cornelis Hoede. 25 years development of knowledge graph theory: the results and the challenge. *Memorandum*, 1876(2):1–10, 2008.

[33] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

[34] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 283–292, 2014.

[35] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. Meta-graph based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 635–644, 2017.

[36] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Dkn: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*, pages 1835–1844, 2018.

[37] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195, 2015.

[38] Kurt Bollacker, Patrick Tufts, Tomi Pierce, and Robert Cook. A platform for scalable, collaborative, structured information integration. In *Intl. Workshop on Information Integration on the Web (IIWeb'07)*, pages 22–27, 2007.

[39] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[40] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard De Melo, and Gerhard Weikum. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World wide web*, pages 229–232, 2011.

[41] Wayne Xin Zhao, Gaole He, Kunlin Yang, Hong-Jian Dou, Jin Huang, Siqi Ouyang, and Ji-Rong Wen. Kb4rec: A data set for linking knowledge bases with recommender systems. *Data Intelligence*, 1(2):121–136, 2019. doi: 10.1162/dint_a_00008. URL https://doi.org/10.1162/dint_a_00008.

[42] Open Source Initiative. The open source initiative, . URL http://opensource.org. last visited on 2020-08-22.

[43] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[44] Open Source Initiative. The bsd 3-clause license, . URL http://opensource.org/licenses/BSD-3-Clause. last visited on 2020-08-22.

[45] NumPy Developers. Numpy license. URL http://docs.scipy.org/doc/numpy/license.html. last visited on 2020-08-22.

[46] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[47] Package overview - pandas 0.14.0 documentation. URL http://pandas.pydata.org/pandas-docs/stable/overview.html. last visited on 2020-08-22.

[48] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

[49] The TensorFlow Authors. Tensorflow license. URL https://github.com/tensorflow/tensorflow/blob/master/LICENSE. last visited on 2020-08-22.

[50] M. González-Fierro A. Argyriou and L. Zhang. Microsoft recommenders: Best practices for production-ready recommendation systems.