Google Summer of Code

Knowledge Graph aware Recommendation System with DBpedia - GSoC 2023*

May 10, 2023

Name Alvaro José Lopes

Email alvarojoselopes@usp.br

Github github.com/AlvaroJoseLopes

Phone +55(35)98877-0098

Location Brazil

Abstract

DBpedia proves to be a way of "Bringing Order to the Web" with structured information as Open Knowledge Graph, contributing towards making quality data more accessible. The main goal of this project is to explore DBpedia's potential at enriching the data provided to Recommender Systems (RS) on different standard datasets, such as MovieLens and LastFM. A framework will be implemented for running reproducible experiments with only the model implementation and a simple .yaml configuration file. Through the framework, this project allows practitioners to easily evaluate and compare their proposed RS algorithms with existing approaches, enabling future benchmarks on enriched and non-enriched datasets. With access to enriched standard RS datasets sourced primarily from DBpedia, this project aims to demonstrate DBpedia's applicability to RS area and other ML areas, potentially promoting DBpedia's adoption and increasing its active community. The main steps of the project are:

- **Entity linking**: between DBpedia and standard RS datasets.
- **Data Enriching**: Build SPARQL queries to enrich RS datasets with useful DBpedia's properties.
- **Framework Implementation**: for reproducible experiments.

*DBpedia forum discussion

Contents

1 The Project

Recommender systems (RS) are designed to provide personalized recommendations by estimating user preferences on unseen items. To achieve this goal, RSs consider previous user interactions and ratings, as well as **additional item and user features, to provide personalized and contextually relevant recommendations**. This may include suggesting similar items based on the user's interests, complementary items to the ones the user has already shown an interest in or recommendations based on user demographics, such as age, gender and location. RSs have been widely studied and applied in various domains, ranging from e-commerce (e.g. Amazon) to multi-media (e.g Netflix, Spotify).

As shown by [22] some knowledge-aware RS use only side information from the original dataset as context data. For example, Point of Interest (POI) information of the Yelp dataset is used in [7] and movie features information of MovieLens dataset (user-generated tags and movie genres, only) are used on the [20] work. However, these datasets usually contain **few kinds of side information** besides user interactions, **failing to provide enough contextual information**.

With the Big Data's Era, a lot of information has become available for all kinds of Machine Learning (ML) tasks, including Recommender Systems, but few applications really exploit their potential. Within the W3C Linking Open Data (LOD) community effort, a lot of data providers started structuring worlds knowledge into Knowledge Bases, such as DBpedia [2], Wikidata [19] and Freebase [8]. On the RS field, there are some works that linked RS datasets with existing KBs, such as DBpedia [18, 11] and Freebase [22].

However, Freebase API has been shut down and the **number of linked datasets is usually small**. For example, [18] linked only 3 datasets (MovieLens, LibraryThing and LastFM). In this scenario, **an open source initiative combining DBpedia and Google Summer of Code efforts** can provide a **greater support**, through the constant effort of the **open source community**, enabling a **broader dataset coverage**. Furthermore, the availability of standard RS datasets enriched with quality data from DBpedia can not only **advance research in this field**, but also **showcase DBpedia as a trustworthy source of data for the RS community and other ML communities**. This project aims towards exploring this potential.

Despite the impressive advancement of ML research, one problem remains unsolved by the ML community: **reproducibility**, a core concept of scientific methodology that enables the **findings** claimed to be **valid and reliable**. Experiments are expected to be repeatable and yield consistent results between runs and reasonable conclusions supported by statistical analysis. However, a Nature's survey [3] showed that over 70% of researchers weren't able to reproduce another researcher's experiments and more than 50% failed to reproduce their own experiments. In the ML research field, some of the reasons for this problem are [13]:

- Lack of access to the same training data and differences in data distribution;
- Under-specification of the model or training procedure;
- Unavailability of the code to run the experiments or code errors when running;
- Under-specification of the metrics used to report results;
- Improper use of statistics to analyze results and others.

The RS community has been discussing it, and concerns about unreproducible evaluation and unfair comparisons have been raised [17]. Various proposed recommendation models have been compared with suboptimal baselines [14] and one survey shows that only 1/3 of published results are reproducible [4].

To address this problem some **frameworks for model evaluation and hyperparameter tuning** were proposed, such as Elliot [1] and DaisyRec [16]. The main goal of this framework is to **provide an intuitive interface**, such as a Command-Line Interface or configuration files (e.g., YAML), that **enables users to easily configure an entire experiment pipeline**.

A theoretical study was conducted by exhaustive literature review to extract and summarize **hyper-factors** that affects RS performance through the pipeline [16]. The hyper-factors were classified into **model-independents** (e.g Dataset pre-processing, splitting method and metric) and **model-dependents** (e.g loss function design, parameter initializer and hyper-parameter tuning strategy). Both frameworks enable hyper-factors configuration, but Elliot is the only one that includes statistical hypothesis on the pipeline, providing paired t-test and Wilcoxon test. Both of them have implemented some popular baselines for comparison.

As **this project aims** to enable future benchmarks on enriched and non-enriched datasets, one of the biggest focuses will be into **providing an extensible and adaptable framework** that **enables users to easily add their own methods** into the framework. That way, practitioners can easily **evaluate** their Knowledge-aware Recommender Systems **with existing approaches** in a reproducible manner, being able to **do their claims more reliably**. At this step, the **open-source community effort** can also provide greater support of this framework, by **implementing more RS model baselines and enabling more hyper-factors configurations through configuration file**.

To accomplish this project's objectives the main steps will be:

- **Data Integration**: the goal of this step is to integrate DBpedia knowledge into RS datasets. To do this each module will be responsible for:
  - **Entity Linking:** identify the corresponding DBpedia entity (URI) for each item in a RS dataset.
  - **Data Enriching:** build SPARQL queries to enrich those RS datasets with useful information for each dataset's domain as item property extracted from DBpedia. Some additional information are: item attributes, related items and extended item attributes.
- **Framework Implementation:** this step aims to implement the framework that enables reproducible experiments into the enriched/non-enriched datasets. The main modules of the framework will be:
  - **DataLoader:** responsible for preparing the data for the model by loading, pre-processing, and splitting into training, validation and test sets.
  - **Recommender:** aims to train the specified model and return the recommendations.
  - **Evaluator:** evaluates the model performance in some metrics.
  - **Reporter:** reports the experiment, providing performance results for each model and recommendation list, with explanations from the supported ones.

For the project quality of life, all **necessary steps for setting the configuration file** to run experiments and **instructions to add a new recommendation model**, for example, will be **documented**. A **leaderboard** will be added to the Github Repository with a pull request template for adding other RS models, that requires: a configuration file for reproducibility, a paper that explains the model, code with model implementation as a subclass of the main *Recommender* module class and experiment results provided by the *Reporter* module.

Another concern of this project is to **highlight explainable RS**. Explainable recommendations are expected to help users make better decisions and improve reliability, effectiveness, persuasiveness and transparency [15]. A fair comparison between models that are

explainable-focused will be done with a separated leaderboard. In this context, the **DBpedia's role** would be essential by providing wealth data to **improve model explainability** and **transparent recommendation**. Explainable RS will be classified into model-agnostic and model-intrinsic approaches.

The Figure 1 shows the project pipeline. The following sections explain in more details each project's step. While the following examples will use the MovieLens dataset as a reference for simplicity, it's important to note that the proposed project pipeline used is suitable for other datasets in the field of RS as well.

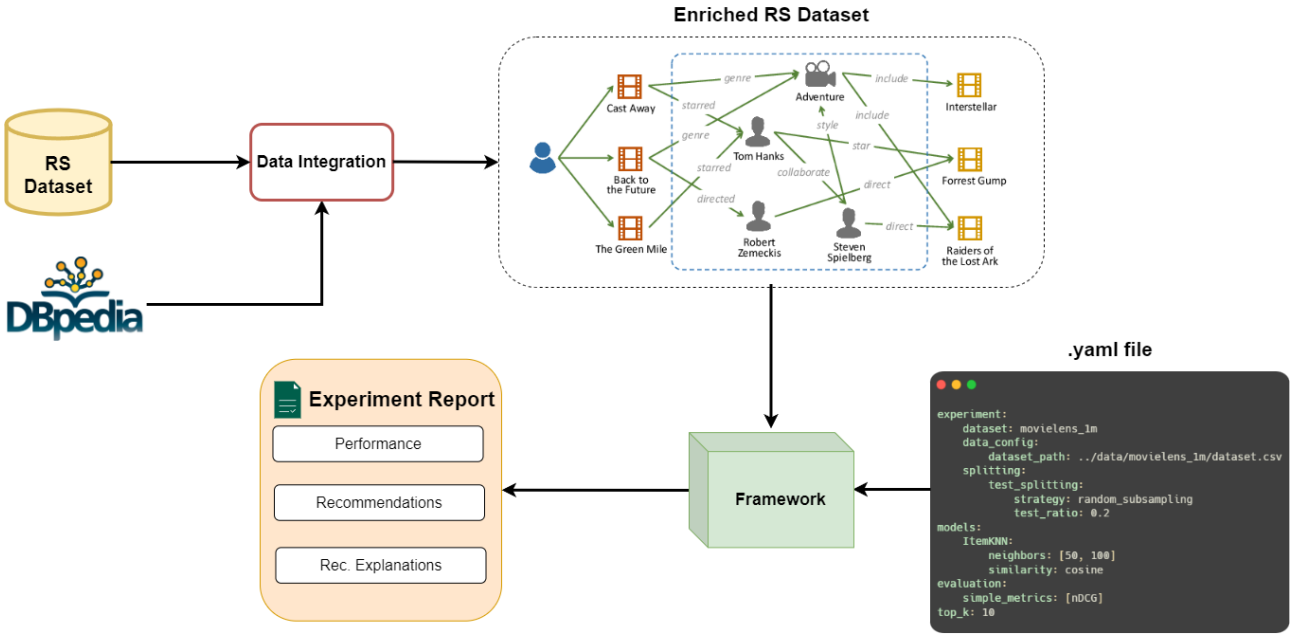

Figure 1: Project proposal pipeline. Made by the author.

1.1 Data Integration

The goal when integrating DBpedia with a RS dataset is to **create a semantic network** that relates items with features and items with other items, providing more knowledge for the RS model, besides user feedback. With **DBpedia** we have access to a huge amount of **factual knowledge of a variety of topics**. For example, the following SPARQL query 1 will return all the properties and values of the Toy Story (1995) movie. This query returns some information about the movie genre, producer, runtime, writers, actors, abstract and others.

```
PREFIX dbr: <http://dbpedia.org/resource/>

SELECT ?property ?value WHERE {
dbr:Toy_Story ?property ?value .
}
```

Source Code 1: A SPARQL query to retrieve all properties and values of Toy Story entity.

Note that for building this query was necessary the URI of the Toy Story film (**dbr:Toy_Story**). Then, for a complete Data Integration, the **first step** is to **map items** described in each dataset (e.g movies, artists, songs, books and products) **to their corresponding DBpedia URI**. This task is called **Entity Linking**.

For each linked entity, a SPARQL query will **retrieve the main properties** about the entity, given their topic. For example, information retrieved for movies will be different from music and products. The main goal of this **second step**, called **Data Enriching**, is to **return a Knowledge Graph** representing the enriched RS dataset **with useful side information (items properties) for RS models**.

The figure 2 illustrate each step of **Data Integration**.

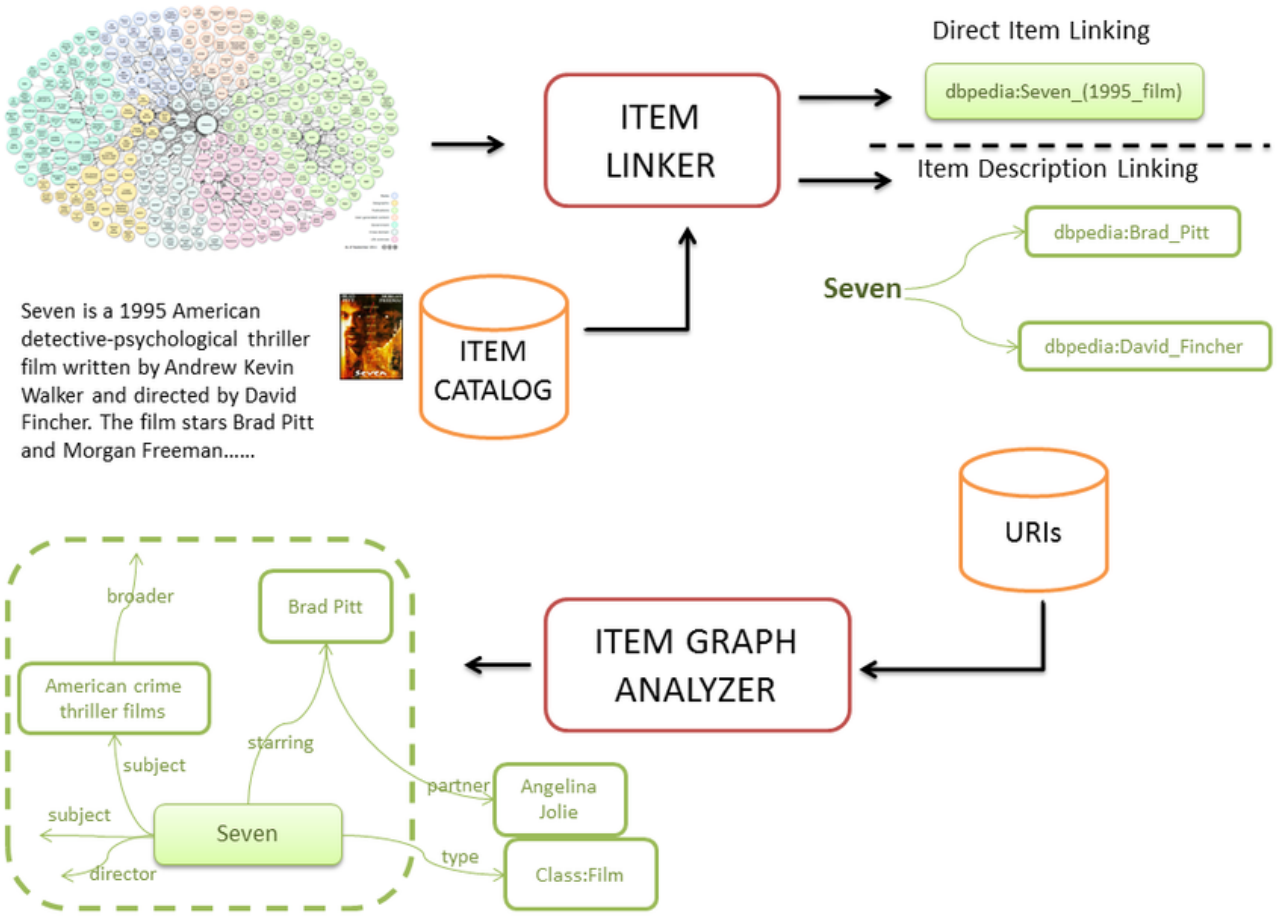

Figure 2: High-level pipeline for **Data Integration** between RS dataset and Knowledge Databases. Made by [5]

1.1.1 Entity Linking (Item Linker)

More precisely, the main task of Entity Linking is to **disambiguate the mention of an entity** e, present in the knowledge base KB, in a text. The final goal is to **map** e **to each mention on the text** [10]. For obtaining better results on this task, is necessary richer texts. However, this is not the case of most RS datasets, where the **only available text is the item name**, like "Toy Story (1995)" on MovieLens.

For retrieving the URI of a dataset's item on DBpedia, a SPARQL query will be used. Given the problem mentioned, the goal is to use all the available information on the dataset to **reduce the number of URIs retrieved by the query**. For example, on the MovieLens dataset, the item name provides the release year of a movie, enabling the use of *dbc:XYZT_films* dbpedia category, where *XYZT* is the year of release, to reduce the search space (e.g 1995_films for Toy Story). The IMDB id is also available on MovieLens, allowing the use of imdbId property. It's also possible to use the type of an item on the dataset from DBpedia ontology, to retrieve

the correct one, when multiples items belong to different domains: for example *dbo:Film* for MovieLens dataset and *dbo:Sound* for LastFM.

The following SPARQL query 2 is an adaptation of [18] and retrieves the URI of "Toy Story (1995)" movie of MovieLens Dataset. It **uses a regular expression to match** the name of the movie and uses *dbc:1995_films* to firstly filter the results. Moreover, as noticed by [18], some entities were not appropriately reached because of the Wikipedia redirection system. For example, "Matrix 2" should match "The Matrix Reloaded", the real name of the second saga movie and a more appropriate DBpedia entity. For doing that the UNION clause uses dbo:wikiPageRedirects property to retrieve the correct entity.

```
SELECT DISTINCT ?film WHERE {
  {
   ?film rdf:type dbo:Film .
   ?film dct:subject dbc:1995_films .
   ?film rdfs:label ?label .
   FILTER regex(?label, "toy.*story", "i")
  }
  UNION
  {
    ?film rdf:type dbo:Film .
    ?tmp dbo:wikiPageRedirects ?film .
    ?tmp dct:subject dbc:1995_films .
    ?tmp rdfs:label ?label .
    FILTER regex(?label, "toy.*story", "i") .
  }
}
```

Source Code 2: A baseline SPARQL query to retrieve "Toy Story (1995)" DBpedia URI.

This is only one example of a possible approach to the problem, and it depends on each RS dataset. It is necessary to devote **some time to study the specificities of each RS dataset and improve the current baseline query**. The Project Timeline specifies when and for how long time will be dedicated to this challenge.

1.1.2 Data Enriching (Item Graph Analyzer)

Once the Entity Linking step is done, a collection of URIs of items will be available. In this next step, the goal is to **extract useful properties from DBpedia's local subgraph for each URI**, by performing SPARQL queries. The previous query 1 returns all properties of the film, but not all of them are useful to relate common preferences of different users and can cause *Curse of Dimensionality*. Then, the next challenge is to **filter the query results by considering the most important properties**.

As analyzed by [10], some properties are more informative than others. In their work, the most informative paths were extracted from MovieLens, LibraryThing and LastFM, by fitting an RF model on the training data and using correspondent variable importance. They saw that the most important ones involve the *dcterms:subject*, *dbpedia-owl:genre*, *dbpedia-owl:wikiPageWikiLink* and, in some cases, *rdf:type*. They believe that those properties connect items with other items belonging to topics or categories semantically related. Furthermore, *wikiPageWikiLink* leverages particular and broader connections, like "Adele" and "Coachella

Valley Music and Arts Festival" that enable finding other artists who performed at Coachella as well.

The properties obtained can be classified in [18]:

- **Item attributes:** for example, genre, director and actors of a movie.
- **Related items:** e.g, *dbo:basedOn* for retrieving the novel that a movie is based on and *dbo:previousWork*/*dbo:subsequentWork* for retrieving prequel/sequel of an item.
- **Extended item attributes:** that needs more than one hop on the ontology to appear as metadata of one item, e.g, the subgenres of a particular music genre with *dbo:muicSubgenre*.

It's worth noting that some time of the project will also be devoted to searching for and analyzing other properties that can provide useful information as input to RS models.

After obtaining useful properties about items, the **Knowledge Graph corresponding to the enriched RS dataset can be built**. The following figure 3 represents a local subgraph of an enriched Knowledge Graph constructed using MovieLens dataset.

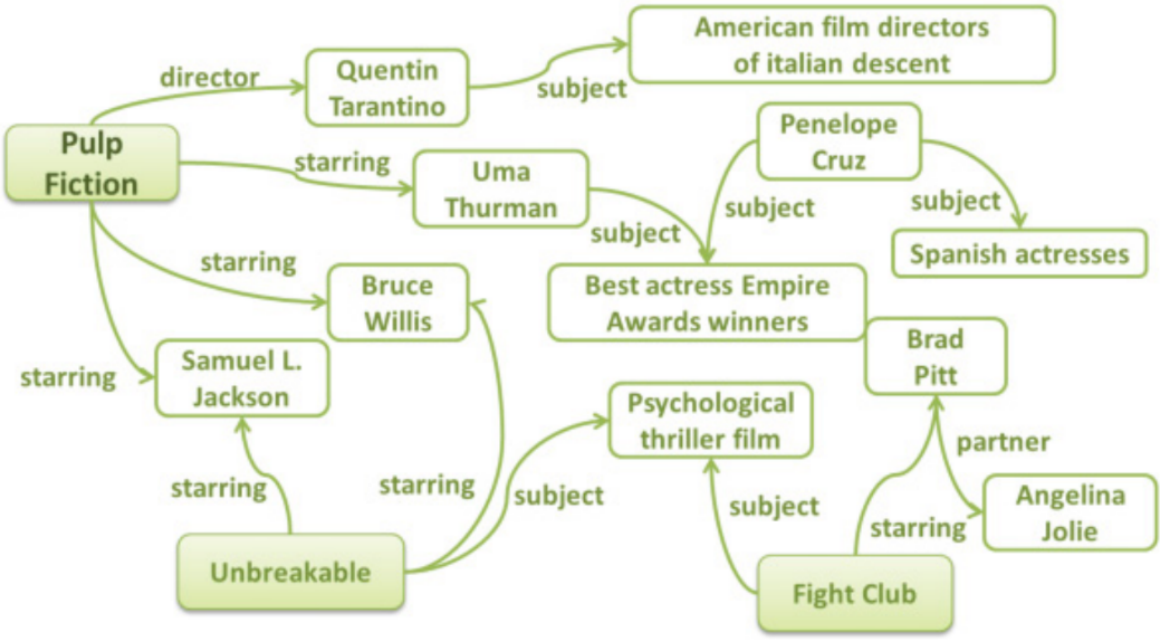

Figure 3: Local subgraph of Knowledge Graph extracted from enriched MovieLens. User's interactions are hided. Made by [10]

1.2 Framework

To facilitate future benchmarking to analyze the impact of enriched data with DBpedia on Knowledge-aware Recommender Systems, **a framework to enable more reproducible experiments** will be implemented. Following [16, 1] ideas, the framework will be **easy to use, extensible and adaptable, where new RS models can be easily added**, enabling open source contributors and researchers to experiment their own methods. In addition, the framework will be **designed to support further contributions** and to maintain a **leaderboard** for tracking the performance of RS models. **The user will be able to configure the entire experiment pipeline with a .yml configuration file**.

The hyper-factors extracted and summarized on the theoretical study [16] will guide the design of the framework, where **each module will be responsible for some hyper-factors**.

The most important ones, as indicated by the study, will be prioritized since the project's timeline is limited. However, **the framework's extensibility will enable more hyper-factors coverage with future open source contributions**.

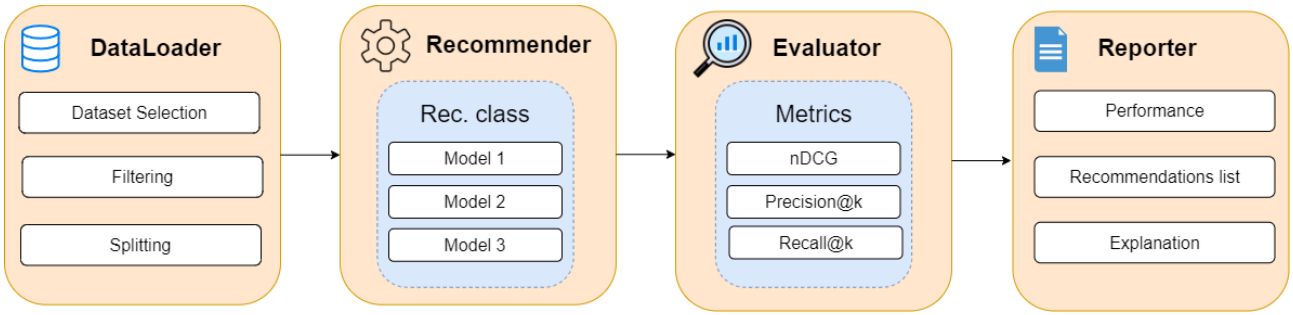

Figure 4: Diagram representing the main modules of the proposed framework. Made by author.

Figure 4 illustrates each module of the framework and figure 5 is a configuration file example.

1.2.1 DataLoader

This module is **responsible for preparing the desired dataset by loading, pre-processing, filtering, and splitting** into training, validations and test sets. The **output of this module will be a Knowledge Graph** as `nx.MultiDiGraph` class from networkx package.

```
experiment:
    dataset: movielens_1m
    data_config:
        dataset_path: ../data/movielens_1m/dataset.csv
    splitting:
        test_splitting:
            strategy: random_subsampling
            test_ratio: 0.2
models:
    ItemKNN:
        neighbors: [50, 100]
        similarity: cosine
evaluation:
    simple_metrics: [nDCG]
top_k: 10
```

Figure 5: Configuration file example.

There are multiple ways of filtering and splitting the data and this project will prioritize the most popular ones as indicated by [16]. DataLoader module deals only with model-independent hyper-factors. The main steps of the experiment pipeline covered by the DataLoader are:

- **Data Selection:** Responsable for **selecting the dataset that will be used for the experiment**. For example, choosing between MovieLens, LastFM and LibraryThing, enriched or non-enriched.

- **Pre-Processing: The feedback of the user can be binarized**. If *binarize flag* is passed as True, the explicit user feedback for each item will be transformed into implicit. For a given threshold r a rating less than r will be considered negative feedback and greater ones will be considered positive feedback.
- **Filtering:** Commonly the standard datasets are sparse, where some users interact only a few times and some items are few interacted. **This submodule filters the inactive users and items**. For example, users must interact with at least K items and each item should be interacted by at least X users.
- **Splitting:** The commonly used methods are split-by-ratio, leave-one-out and split-by-time.
  - **Split-by-ratio:** proportions $train_{size}$ and $validation_{size}$ and $test_{size}$ are used to split all the data into training, validation and test data.
  - **Leave-one-out:** for each user, only one record is kept as test data, and the rest are for training and validation data.
  - **Split-by-time:** splits the data into train, validation and test data based on interaction timestamp.

1.2.2 Recommender

This module **provides a Python class for each implemented RS model**. All classes of models are a **subclass of** `Recommender` main class and require the implementation of `train()`, `get_recommendations()` and `get_single_recommendation()` methods. Those methods are the main interfaces for training and evaluating the model:

- `train()`: defines the training strategy of the proposed RS model.
- `get_recommendations()`: returns all the first k predicted recommendations in order for validation and test data.
- `get_single_recommendation()`: returns a ranked list of k recommendations for a given user.

An optional method is `get_explainable_recommendation()`, which returns a ranked list of recommendations with their explanations for a given user. With the framework, the user can **set all the specific parameters of the model**, e.g number of neighbors of ItemKNN, **through the configuration file**.

To show the framework's capability a **node-embedding Recommender System baseline will be implemented** [21]. The first step of the baseline will use a **Node embedding** technique to learn user and item embeddings, then with the learned vectors the final step will be **Ranking** the user preference, by the similarity between user and items or by Link Prediction. The pipeline of the baseline is:

1. **Node Embedding**: Node embedding is a technique for transforming nodes in a graph into dense vectors in a low-dimensional space. The goal of node embedding is to capture the underlying patterns and relationships between nodes in the graph, which can be useful for various tasks, including recommendation systems. There are several node embedding techniques available, such as DeepWalk [12] and Node2Vec [9]. The goal of this step is to **learn user and item embeddings in a way that users are closer in the embedding space to the items they prefer**. In the case of enriched datasets, the item's properties will influence the resulting embeddings and will reflect the underlying relationships between items that share similar properties.

2. **Ranking**: Ranking items in a recommender system involves **presenting the top-k items recommendations for a user**. The output of this step is a list of items, sorted in descending order of predicted relevance or preference for the user. With the embeddings obtained, there are some possible approaches for ranking, such as using **similarity measures** or performing **Link Prediction**.

   (a) **Similarity measure:** When using similarity measures the goal is to **calculate the similarity between a user and a set of items**, and then **use this similarity score for ranking** recommendations. In this context, one commonly used similarity measure is Cosine Similarity.

   (b) **Link Prediction:** The goal of link prediction is to **infer the existence of a link or relationship between a user and an item** that the user has not yet interacted with. There are several approaches to perform Link Prediction, including using deep learning-based classifiers or a Logistic Regression. This project will use a simple **Logistic Regression as baseline**. The Logistic Regression algorithm **models the probability of a binary outcome as a function of the input features**. In the context of RS, the input will be the item and user embeddings, while the binary outcome will represent whether the user will interact with the item or not. The final step is to **sort the items in descending order of their predicted probability scores** and present the top items to the user as recommendations.

Due to the limited time of the project, this model will be the only one implemented. Worth noting that **future contributions with other model implementations will be supported**, which will enable future benchmarks on enriched and non-enriched datasets.

1.2.3 Evaluator

This module is **responsible for evaluating the RS models**. Through this module the **user can choose any metric to drive the model tuning and evaluation**. There are a lot of metrics for Accuracy, Error, Coverage, Novelty, Diversity, Bias and Fairness [1], but the project will emphasize the most commonly used metric for measuring ranking quality: Accuracy. For this project, the metrics to be implemented are:

- **nDCG:**[1] extends for normalized Discounted Cumulative Gain and measures the information gain obtained for each recommendation based on its position in the result list. Discounted refers to the fact that the discount on information gain is greater for lower ranks recommendations. The values are normalized between 0 and 1.

- **Precision@k:**[2] calculates the number of correct recommendations by the total number of recommendations. The value of k indicates the top-k recommendations for the user. Let Rel_u be the set of relevant items and Rec_u^k the set of top-k recommendations for the user, $Precision@k$ is defined as:

$$Precision@k = \frac{|Rel_u \cap Rec_u^k|}{|Rec_u^k|}$$

  The final result is the average for each user.

[1] nDCG wikipedia
[2] Precision wikipedia

- **Recall@k:**[3] calculates the number of correct recommendations by the total number of relevant items for the user. $Recall@k$ is defined as:

$$Recall@k = \frac{|Rel_u \cap Rec_u^k|}{|Rel_u|}$$

As before, the final result is the average for each user.

1.2.4 Reporter

When the experiment is finished, a **report of the results** will be available. The report includes the output of these submodules:

- **Performance:** responsible for making available a .csv file with all the metrics computed for each model on the test set. If more than one parameter is explored, then each combination is reported.
- **Recommendations list:** the goal is to store, on a .csv file, the top-k recommendation lists for each user made by all the models evaluated.
- **Explanation:** provides the model explanation, from the supported ones, for each recommendation given to a user.

1.2.5 Future Contributions

To keep track of the model's performances, a leaderboard will be created and **newly proposed models can be included on it by a pull request to the repository**. To conclude the pull request, certain template information is required:

- Full implementation of the model as a subclass of `Recommender` main class;
- A summary of the model with respect to some general information that enables classifying the models. E.g classify into explainable and non-explainable models.
- Link to a paper explaining the model;
- .yml configuration file to reproduce the experiment;
- A copy of the **Performance** report from the **Reporter**.

The leaderboard can be improved with new contributions, improving the baseline's coverage and encouraging newly proposed methods to get a good ranking. As a way to encourage the open-source and Knowledge-aware RS communities, **this project will suit well to new GSoC contributors, future benchmarks and challenges**, like BiKE (First International Biochemical Knowledge Extraction Challenge)[4], an initiative originated from NatUKE (Natural Product Knowledge Extraction) benchmark [6].

This framework **lowers the barrier to entry for contributors**, as they do not need to worry about data manipulation and pre-processing (DataLoader handles that) or model evaluation (Evaluator is responsible for that). **Contributors only need to provide the model implementation**.

[3] Recall wikipedia
[4] BiKE challenge site

1.3 Benefits

This project can demonstrate the value of DBpedia by showing **how DBpedia can be used to enhance the quality and explainability of Recommender Systems**. Furthermore, a broader research community can be reached, encouraging **more researchers to use DBpedia in their work**. This can also **increase the active community**, bringing new contributors to DBpedia.

The framework will provide an easy-to-use interface for researchers to configure and run experiments on enriched datasets. This will **make it easier for researchers to use DBpedia in their Recommender Systems research**, without having to go through the tedious process of linking DBpedia data to their datasets. This can potentially lead to a more **widespread adoption of DBpedia in the field**.

Furthermore, the enriched datasets and framework can be used for **benchmarking and comparing different algorithms**, which can help establish best practices and standards for Recommender Systems research.

In the end, this project can also **foster more collaborations between the RS and DBpedia communities**, leading to the development of new tools, techniques, and best practices that benefit both communities.

2 Project Timeline

I have no prior commitments and I will be able to devote 30-35 hrs per week.

2.1 Community Bonding Period (until 28th May)

- Get to know my mentors and learn a lot from them.
- Discuss the project with my mentors in further detail.
- Get more acquainted with DBpedia, Recommender Systems, Open source standards and ML frameworks for reproducibility. And with that, expand my references.
- Read the bibliography references in further detail.
- Try to propose new ideas to improve the baselines presented in each project step.

2.2 Coding Period

The following table 1 summarizes which weeks will be dedicated to each Activity and Framework Module. Worth noting that while each module is implemented, it will be documented.

Table 1: Summarized timeline for Coding Period per Activity and code Module.

	Weeks													
Activities and Modules	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Entity Linking	x	x	x											
Data Enriching				x	x									
Preparing Framework						x	x	x						
DataLoader						x	x	x						
Recommender									x	x				
Evaluator											x	x		
Reporter											x	x		
Wrap-up													x	x
Repository documentation	x	x	x	x	x	x	x	x	x	x	x	x	x	x

The following sections detail each week's activities.

2.2.1 Week 1, 2 and 3 (29th May - 18th June)

This period will be devoted to the **Entity Linking step** for each dataset.

1. Searching for standard RS datasets that can be integrated with DBpedia (at least 3).
2. Studying each dataset specificity, while looking for ways to link each one with DBpedia URIs.
3. Applying the Entity Linking baseline [18] and evaluate the results (e.g, dataset coverage and match quality).
4. Analyzing the initial results and looking for ways to improve the task on each dataset.

2.2.2 Week 4 and 5 (19th June - 2th July)

The goal of this period of coding is to complete the **Data Enriching step** for each dataset.

1. First, using the SPARQL query 1 to retrieve all the properties of some items from each dataset and also retrieve some broader properties (properties in more depth, e.g, 1+ hops).
2. Analyzing the properties retrieved and identifying the most important ones, using [10] as reference.
3. Building a final SPARQL query for each dataset to extract the most useful properties.
4. Finally, integrating the RS dataset and DBpedia extracted information into a Knowledge Graph as `nx.MultiDiGraph`.

2.2.3 Week 6, 7 and 8 (3th July - 23th July)

This period will begin the framework implementation and aims to **get the framework repository and the initial structure of the framework ready**. Also aims to **implement the DataLoader submodule** to enable loading the non-enriched/enriched chosen datasets.

1. Getting ready the framework repository by preparing the folder structure considering the modules proposed.

2. Preparing a module to deserialize YAML into a Python dictionary to retrieve the parameters passed by the user.
3. Implementing the Dataset Selection submodule with respect to the datasets chosen in previous steps.
4. Choosing and implementing the main filtering techniques, using [16] as a reference in the choice, and implement them into a submodule.
5. Choosing and implementing the main splitting techniques, also taking [16] as a reference in the choice and implement them into a submodule.
6. Supporting all the DataLoader features through the .yml file.

2.2.4 Week 9 and 10 (24th July - 6th August)

This period aims to **implement the Recommender submodule**.

1. Implementing the `Recommender` main class.
2. Implementing the Recommender System baseline proposed on Recommender section.
3. Supporting the model and its parameters through the .yml configuration file.

2.2.5 Week 11 and 12 (7th August - 20th August)

This period will be used to **implement the Evaluator and Reporter** modules.

1. Implementing the metrics: nDCG, Precision@k and Recall@k on Evaluator module.
2. Supporting all metrics into .yml file.
3. Implementing the Performance submodule.
4. Implementing the Recommendation list submodule.
5. Implementing the Explanation submodule.
6. Supporting multiple reports for a model if more than one configuration was provided.
7. Add all submodule features into a main class that returns the entire report of the experiment.

2.2.6 Week 13 and 14 (21th August - 4th September)

This period will be devoted to **wrapping up the project**.

1. Minor changes to improve the project.
2. Final testing of an entire experiment pipeline using the Data Integration and the Framework.
3. Implementing fixes.
4. Cleaning up and refactoring the code.
5. Final documentation of the repository for framework's usage and future contributions.

2.3 Post GSoC

I hope the project went well and the most important **objective** was accomplished: **making the project well suited for future open source contributions** from other GSoC contributors, DBpedia and Knowledge-aware RS communities. I also hope that this project will **encourage me to contribute more to the open-source community**.

I believe that there are a lot of other ML areas, besides Recommender Systems, that would benefit from a combined effort between GSoC and DBpedia, and I hope that I somehow contributed to it and can **contribute more** in future works. With this, I hope to be able to **strengthen ties** with DBpedia and my mentors.

3 About Me

3.1 Background & Education

I am in my 4th year pursuing a Bachelor in Computer Science at the Institute of Mathematics and Computer Science, University of São Paulo, Brazil. I expect to graduate in January 2025. I am also a Research Assistant and my research interests are Machine Learning applied to Graphs, Natural Language Processing, and Explainable AI. Currently, I'm working on Graph Neural Networks (GNNs).

I am a member of Data-ICMC, an extracurricular group managed by students interested in Data Science and Machine Learning. At Data-ICMC, I worked on some projects about Complex Networks Analysis and Time Series Forecasting, and taught some lessons, with the support of jupyter-notebooks for homework practice, about Statistics and Natural Language Processing (NLP) for college newcomers.

3.2 Technical Skills

I am comfortable using:

- **C/C++**: It's the first programming language that I learned and have been using since 2018. I code in these languages mainly for more low-level projects at University.
- **Python**: Have been using it since 2019 and it's my main programming language. I use Python for developing Machine Learning and Data analytics projects. Some projects that I developed were designed as easy-to-use Python packages, inspired by scikit-learn.
- **Linux**: Have been using it since 2020 and Linux has improved my workflow since then.
- **LaTeX:** Have been using it since 2020 and LaTeX has provided extremely good control over the formatting for my reports on university and for my research.

3.3 Past projects

These are some projects about Data Science and Machine Learning that I developed:

- **DeepWalkTransformers:** An easy to use **Python API** for **Node Embedding**. We are using a **NLP-based** approach that uses **Masked Language Modelling** (**BERT**) for learning transductive/inductive embeddings. The model also supports **feature information**, besides graph structure. My goal was to make a **clean**, **scalable**, **failsafe** and **extendable** code, that is **inspired on the principles** used to create the widely used **scikit-learn package**. This model is part of my research project.

- **github-repo-graph:** In this project, the intention was to **transform the dependency relationships between github repositories into a graph**. Then, perform **exploratory data analysis**, **visualization** and **data mining** to extract some knowledge of the data. In this project, I was able to learn about **Web Scrapping** and **GraphQL queries** to acquire data from the GitHub API, as well as introductory knowledge about **complex networks**. At the end of the project, I wrote a **blogpost** (in Portuguese) to clearly explain the project.

- **Text Vectorization:** **Word Embedding** baseline implementation based on PPMI matrix factorization. We also explored the embedding space learned using **dimensionality reduction**, **hierarchical clustering** and **analogy similarity**.

- **DATA-ICMC teaching:** I built some jupyter-notebooks for students of the classes promoted by DATA-ICMC, so they can **practice the theory seen in class** and **get familiarized with the most used Data Science libraries**. I was responsible for building the slides of the **NLP class** that I taught and for building the homework notebooks of NLP and **Regressions class**. The content is in Portuguese.

3.4 Open Source

This will be my first time contributing to an open-source project. As a heavy user of open source projects, I believe it's **time to give back to the** community, and not only to the **open-source community** but also to the **Machine Learning community**.

Because this project will create a new project repository, not contribute to an existing one, I believe it will be a good opportunity to **understand how an open-source project is structured**, how the project is **designed to enable future contributions** and how to **support** the project. And the best: learn this in a practical way with my project.

3.5 Summer Plans

- **What city/country will you be spending this summer in?**
São Carlos, São Paulo, Brazil.

- **Do you have a full- or part-time job or internship planned for this summer?**
No, I don't.

- **How many hours per week do have available for a summer project?**
25-30 hours per week.

3.6 GSoC Experience

- **Did you participate in a previous Summer of Code project?**
No, I didn't.

- **Have you applied or do you plan to apply for any other 2023 Summer of Code projects?**
This project will be the only one that I plan to apply.

- **Why did you decide to apply for a DBpedia project and why this project?**
I decided to apply to a DBpedia project because I believe DBpedia has a lot of untapped potential to contribute to **Machine Learning community**, such as the Recommender System. It's fascinating to see how these ML projects **benefit from the knowledge**

provided by the DBpedia Knowledge Graph. DBpedia proves to be, not only a dataset but an association that really **supports the community and encourages new projects**.

There were a lot of interesting projects for DBpedia, but I chose this one because Knowledge Graphs are one of my **research interests** and would be a great **opportunity to apply what I'm learning in my research**. Working with Recommendation Systems would be a way to **expand my knowledge**, since I never worked with it.

- **Why me?**
  As pointed out before I'm already **familiar with some subjects that are necessary** to complete the project and since I'm not touching completely unknown ground, I believe this will **facilitate my integration into the project**.

  I believe my project proposal is **well structured** and shows that the project was **well planned**. It's worth noting that I was not worried only about During-GSoC time, but was also thinking about Post-GSoC and about the **project in long-term**. The fact that I'm proposing to build a project that **supports future contributions** by other GSoC contributors and researchers, shows that.

  Given all that, **I believe I'm a good fit with a good proposal for the project**.

Worth noting that I would be happy to make any changes required by the mentors to make this proposal/project better.

4 Bibliography

[1] Anelli, V. W., Bellogín, A., Ferrara, A., Malitesta, D., Merra, F. A., Pomo, C., Donini, F. M., and Noia, T. D. (2021). Elliot: a comprehensive and rigorous framework for reproducible recommender systems evaluation. *CoRR*, abs/2103.02590.

[2] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., and Cudré-Mauroux, P., editors, *The Semantic Web*, pages 722–735, Berlin, Heidelberg. Springer Berlin Heidelberg.

[3] Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454.

[4] Dacrema, M. F., Cremonesi, P., and Jannach, D. (2019). Are we really making much progress? A worrying analysis of recent neural recommendation approaches. *CoRR*, abs/1907.06902.

[5] Di Noia, T. and Ostuni, V. (2015). *Recommender Systems and Linked Open Data*, volume 9203, pages 88–113.

[6] Do Carmo, P. V., Marx, E., Marcacini, R., Valli, M., Silva e Silva, J. V., and Pilon, A. (2023). Natuke: A benchmark for natural product knowledge extraction from academic literature. In *2023 IEEE 17th International Conference on Semantic Computing (ICSC)*, pages 199–203.

[7] Gao, H., Tang, J., Hu, X., and Liu, H. (2015). Content-aware point of interest recommendation on location-based social networks. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, page 1721–1727. AAAI Press.

[8] Google (2023). Freebase data dumps. https://developers.google.com/freebase/data.

[9] Ma, E. (2019). Random walk in node embeddings (deepwalk, node2vec, line, and graphsage). *Towards AI*.

[10] Noia, T. D., Ostuni, V. C., Tomeo, P., and Sciascio, E. D. (2016). Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Trans. Intell. Syst. Technol.*, 8(1).

[11] Oramas, S., Ostuni, V. C., Noia, T. D., Serra, X., and Sciascio, E. D. (2016). Sound and music recommendation with knowledge graphs. *ACM Trans. Intell. Syst. Technol.*, 8(2).

[12] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652.

[13] Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Larochelle, H. (2020). Improving reproducibility in machine learning research (A report from the neurips 2019 reproducibility program). *CoRR*, abs/2003.12206.

[14] Rendle, S., Zhang, L., and Koren, Y. (2019). On the difficulty of evaluating baselines: A study on recommender systems. *CoRR*, abs/1905.01395.

[15] Shimizu, R., Matsutani, M., and Goto, M. (2022). An explainable recommendation framework based on an improved knowledge graph attention network with massive volumes of side information. *Knowledge-Based Systems*, 239:107970.

[16] Sun, Z., Fang, H., Yang, J., Qu, X., Liu, H., Yu, D., Ong, Y.-S., and Zhang, J. (2022). Daisyrec 2.0: Benchmarking recommendation for rigorous evaluation.

[17] Sun, Z., Yu, D., Fang, H., Yang, J., Qu, X., Zhang, J., and Geng, C. (2020). Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In *RecSys 2020 - 14th ACM Conference on Recommender Systems*, RecSys 2020 - 14th ACM Conference on Recommender Systems, pages 23–32, United States. Association for Computing Machinery (ACM). 14th ACM Conference on Recommender Systems, RecSys 2020 ; Conference date: 22-09-2020 Through 26-09-2020.

[18] Tomeo, P., Fernández-Tobías, I., Di Noia, T., and Cantador, I. (2016). Exploiting linked open data in cold-start recommendations with positive-only feedback. pages 1–8.

[19] Vrandečić, D. and Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

[20] Yu, X., Ren, X., Sun, Y., Gu, Q., Sturt, B., Khandelwal, U., Norick, B., and Han, J. (2014). Personalized entity recommendation: A heterogeneous information network approach. In *WSDM 2014 - Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM 2014 - Proceedings of the 7th ACM International Conference on Web Search and Data Mining, pages 283–292. Association for Computing Machinery. 7th ACM International Conference on Web Search and Data Mining, WSDM 2014 ; Conference date: 24-02-2014 Through 28-02-2014.

[21] Zhang, M. (2022). Graph neural networks: Link prediction. In Wu, L., Cui, P., Pei, J., and Zhao, L., editors, *Graph Neural Networks: Foundations, Frontiers, and Applications*, pages 195–223. Springer Singapore, Singapore.

[22] Zhao, W., He, G., Dou, H., Huang, J., Ouyang, S., and Wen, J.-R. (2018). Kb4rec: A dataset for linking knowledge bases with recommender systems.