Movie Recommendation using Knowledge Graphs

192.116 Knowledge Graphs

Patrick Indri
patrick.indri@tuwien.ac.at

October 30, 2022

1 Introduction

Understanding the preferences of a set of users for a set of items (e.g., movies or books) has become an important task as the web developed and became accessible since the early 2000s. Recommender systems (Bobadilla et al. 2013) use various sources of information to characterise items and interactions between items and users, with the end goal of providing users with pertinent item suggestions. For this project, the focus is on movie recommendation.

Collaborative filtering (Schafer et al. 2007), a commonly adopted technique to tackle the recommendation task, is concerned with evaluating and recommending items by means of the opinion of other users. Collaborative filtering based recommendation has proved to be effective (Schafer et al. 2007; Herlocker et al. 2004) and can utilise a variety of unstructured data such item reviews or ratings. However, it is not well suited to model additional information on relationships between items and users.

Knowledge graph (KG) embeddings (Q. Wang et al. 2017) solve this problem by embedding an heterogeneous set of entities and relationships into a unified space (e.g., a continuous vector space) while preserving the structure of the KG itself. This allows for a richer characterisation of the interactions between users and items and can lead to better recommendation performance.

This project will focus on the task of movie recommendation using KG embedding, providing a comparison between the various embeddings adopted, a matrix factorisation, and a Graph Neural Network-based approach. Additionally, privacy concerns and differentially private KG embeddings will be discussed. (LO11)

2 Background

In this section, a brief outline of the collaborative filtering techniques address in the project is presented. In particular, the focus of the project is on the use KG embeddings for recommendation. Additionally, differential privacy (Dwork 2008) is briefly introduced, as the impact of differential privacy in KG embeddings will also be discussed. The discussion will be framed in the context of movie recommendation.

2.1 Collaborative filtering and matrix factorisation

Consider a *feedback* or *ratings* matrix R, where the position R_{ij} contains the rating (e.g., on a scale of 1 to 5, or as a binary value to denote appreciation for the movie) of user i for movie j, or 0 in case there is no rating. In this context, recommendation aims at predicting the feedback of user i on the set of movies with with i has not interacted yet. Informally, *collaborative filtering* works under the assumptions that two users who share similar taste for a certain set of movies are more likely to have the same opinion on other movies.

More formally, objective is to decompose the ratings matrix R as a low-rank product of two matrices:

$$\underset{m\times n}{R} \approx \widetilde{R} = \underset{m\times k}{X}\,\underset{k\times n}{Y}$$

where U and V are $k-$dimensional embeddings for the m users and n movies.

Solving this factorisation problem leads to the objective function

$$\mathcal{L} = \sum_{i,j} c_{ij}(R_{ij} - x_i^T y_j)^2 + \lambda \left(\sum_i \|x_i\|^2 + \sum_j \|y_j\|^2 \right) \tag{1}$$

where $c_i = \sum_{i,j} 1$ if $R_{ij} > 0$, and x_i and y_j are row vectors of X and Y. This non-convex problem can be solved using the Weighted Alternating Least Squares (WALS) algorithm. Recommendations for user i can then be obtained by ranking its unobserved movies in $\widetilde{R}$.

This approach can be effective and efficient (as WALS can be parallelised), but considers user feedback only.

2.2 Knowledge graph embedding for recommendation

In contrast, KG embeddings can effectively model multiple relationships between users and items. (LO1)
The approach discussed (and implemented, see Section 3.2) here follows closely the one presented in Y. Zhang et al. 2018.

In particular, let us consider a KG constituted by a set of triplets (h, l, t) where h (head) and t (tail) belong to a set of *entities* and l belongs to a set of relationships.

Following the energy-based model *TransE*[1] presented in Bordes et al. 2013, the idea is to learn low-dimensional vector embeddings for the triplets such that the functional relationship between the heads $\boldsymbol{h}$ and tails $\boldsymbol{t}$ corresponds to a translation by the relationships $\boldsymbol{l}$. That is, in the embedding $\boldsymbol{h} + \boldsymbol{l} \approx \boldsymbol{t}$ should hold. The embedding can be learnt by minimising the margin-based loss:

$$\mathcal{L} = \sum_{(h,l,t)\in S} \left[\sum_{(h',l,t)\in S^h} (\gamma + \|\boldsymbol{h} + \boldsymbol{l} - \boldsymbol{t}\| - \|\boldsymbol{h'} + \boldsymbol{l} - \boldsymbol{t}\|)_+ + \right. \tag{2}$$
$$\left. \sum_{(h,l,t')\in S^t} (\gamma + \|\boldsymbol{h} + \boldsymbol{l} - \boldsymbol{t}\| - \|\boldsymbol{h} + \boldsymbol{l} - \boldsymbol{t'}\|)_+ \right]$$

where $\gamma > 0$ is the margin, $[x]_+$ denotes the positive part of x, and S^h and S^t are corrupted triplets obtained by replacing (respectively) the head and tail entity in the training triplets with

random entities. Optimisation can be carried out by *stochastic gradient descent* (SGD).

Recommendations with respect to the relationship of interested can then be obtained by considering the euclidean distance between entities, given the relationship. For instance, considering a *"feedback"* relationship denoting positive interaction between users and movies, recommendations for user u_i can be produced by finding the movies v_j that minimise (LO9) (LO6) (LO11)

$$\|\boldsymbol{u}_i + \boldsymbol{l}_{\text{feedback}} - \boldsymbol{v}_j\| . \tag{3}$$

This approach, following the taxonomy in Guo et al. 2020, is to be considered an *embedding-based* method to recommendation, as the learning process can be divided in two basic modules: an embedding module and a recommendation module based on the previously obtained embedding.

2.3 Propagation based methods for recommendation

While embedding-based methods take advantage of the semantic relationships between items and users, propagation-based methods aim at capturing complex and high-order relationships between the entities as well. The core principle is that of aggregating embeddings of multi-hop neighbours in the KG to refine the embeddings themselves. Often, the implementation is based on Graph Neural Network (GNN) methodologies. (LO3)

For the scope of the project, we consider the Knowledge Graph Convolutional Network (KGCN) proposed in H. Wang et al. 2019. Inspired by Graph Convolutional Networks (S. Zhang et al. 2019), KGNN captures structural proximity of entities in the KG by means of an aggregation procedure where the weight of each neighbour is user specific. The target is a prediction function that, given the knowledge graph and the user-item interaction matrix, aims at predicting the probability that a certain user will positively interact with a certain item. In case of probability larger than 0.5, the user is predicted to interact with the item. The learning task is therefore framed as that of binary classification, where the training data contains both instances where user and item interact and instances where they do not interact, and the testing scenario simply requires to predict whether there will be interaction between any give user-item pair. It should be therefore highlighted that the learning task optimises for low classification error, and not in particular for a ranking in the probability of interaction. A ranking of the predicted items can nevertheless be given by sorting the probability of interaction and recommending the items with higher probability.

2.4 Differential privacy

Differential Privacy (DP) (Dwork 2008; Dwork, Roth, et al. 2014) is a framework which aims at protecting the privacy of records in a dataset by giving plausible deniability of their presence. This is achieved by means of a randomized algorithm $\mathcal{M}$, a mechanism that acts on datasets. Considering two datasets D and D' with the latter differing from the former for a single record (i.e., being a *neighbouring* dataset, denoted as $\|D - D'\| = 1$), DP assures that executing $\mathcal{M}$ over D or D' gives similar results, thus protecting the presence/absence of a record.

More formally, an algorithm $\mathcal{M}$ is (ε, δ)-differentially private if for all $E \subseteq \text{Range}(\mathcal{M})$ and for all D, D' such that $\|D - D'\| = 1$:

$$\Pr[\mathcal{M}(D) \in E] \leq \exp(\varepsilon) \Pr[\mathcal{M}(D') \in E] + \delta$$

[1]The report will consider TransM (Fan et al. 2014) as well. As TransE and TransM share many similarities, with differences mainly consisting in the latter considering pre-computed weights for the scoring function, TransM will not be discussed in detail and considerations presented for TransE hold for both approaches.

where the probability space is over the coin flips of the mechanism $\mathcal{M}$.

Intuitively, this assures that, with high probability, we cannot distinguish the output of our algorithm over two similar datasets and we can thus not say for certain whether a specific record is part of the dataset. In the context of DP, the value ε is referred to as the *privacy budget* of the algorithm.

The concept of neighbouring datasets translates to KG as the concept of *edge neighbouring knowledge graphs* (Han et al. 2022), in which two KG K and K' are edge-neighbour if they differ for one statement, that is, if they differ for a single triplet (h, l, t). (LO12)

The application of DP to KG introduces the idea that two datasets which are similar should produce similar embeddings. Ideally, this could avoid the necessity of, e.g., removing confidential triplets from a KG when learning the embedding.

One way to achieve a differentially private embedding is to perform the optimisation of, e.g., Eq. (2), using the differentially private version of SGD (DP-SGD) introduced in Abadi et al. 2016. Although the details of DP-SGD are out of the scope of this report, one fundamental detail of its inner workings is that it requires gradients to be computed on individual data points and that it assumes that the gradient-based optimisation is the only portion of the overall algorithm that has access to the data. For KG embeddings, this means that the only part of the procedure which can update the embeddings themselves and has access to data shall be the gradient descent step. Embeddings which satisfy such requirements are called *gradient-separable* (Han et al. 2022).

The embeddings on which this report will focus, namely TranE and TransM, are gradient separable and can be thus privatised.

3 Method

This section describes how the knowledge graph used for the project can be obtained and how the various implementations and experiments were carried out. Considerations on the evolution and scalability of the approach are deferred to Section 5.

3.1 Obtaining data and knowledge graph

The MovieLens dataset is a dataset collected by GroupsLens Research which gathers movie ratings for tens of thousands of movies and users. In particular, a reduced version[2] of the full dataset was used for this project. Similarly to what described in Section 2.1, the dataset essentially consists of a matrix of user-movie ratings. Specifically, MovieLens ratings are on a scale from one to five.

To enrich the information present in MovieLens, external sources can be used to obtain metadata information for the reviews, such as the name of the director of the movie the reviews considers. DBpedia (Auer et al. 2007) is a project that makes available structured content extracted from Wikipedia. The properties of the various entities stored in DBpedia are classified by means of a consistent ontology and DBpedia allows for queries of properties and relationship obtained from Wikipedia articles. Specifically, DBpedia adopts the Resource Description Framework (RDF) as a data model to represent the extracted information as semantic triplets. In particular, and similarly to the formalism introduced Section 2.2, the nodes of the graph are the head and tail of the triplet (called *subject* and *object*, in RDF terminology) and the central element of the triple (LO4)

[2]Available at https://grouplens.org/datasets/movielens/latest/.

(*predicate*) is an edge in the KG. Data stored in RDF format can then be queried by means of specific query languages such as SPARQL, which allow for queries to be formulated in terms of triplets. A parallelism can here be drawn to non-relational databases, that is, databases which rely on non-tabular storage of data and which often support data retrieval via queries of the form (document-)key-value. Both RDF and non-relational databases[3] should instead be compared against the more rigid and structured relational databases, which relies on well defined tables and relationships between tables. Despite the better seek time achievable by relational databases by means of indices, particularly at large scales, the unstructured nature of the information contained in Wikipedia would render the definition of a relational schema problematic and highly inflexible to future changes in the DBpedia ontology.

Specifically, mappings from MovieLens to DBpedia were obtained by Ostuni et al. 2013 and used by Palumbo, Rizzo, and Troncy 2017 to formulate SPARQL queries for the DBpedia endpoint. Palumbo, Rizzo, and Troncy 2017 make available the results of the queries, which consist in a series of subgraphs, one for each of the properties (such as director, composer) queried. For this project, the subgraphs were merged into a KG (`data_builder.py`) and indices were generated for the various entities. In particular, a *feedback* relationship is used to denote a connection between a user and a movie when the user has reviewed the movie with four or more stars. The result KG consists of text file containing all the triplets (including the feedback one). In summary, the creation of the KG consists mainly of a data linkage operation between MovieLens and DBpedia, which is much simplified by the fact that entities in MovieLens have been stable for some years and that mappings to DBpedia were made available by Ostuni et al. 2013. (LO7)

3.2 Implementation

The movie recommendation solutions described in Section 2.2 were implemented from scratch in Python. In particular, `models.py` provides a PyTorch implementation of TransE and TransM and `main.ipynb` defines the data loading, training, and testing routines necessary for the recommendation task. (LO1)

`WALS/` contains a Python implementation of the WALS algorithm for matrix factorisation described in Section 2.1. It should be noted that most of this code was produced as a result of another project, and that adaptation were sufficient to apply it to this task and dataset.

`KGCN/` contains an implementation of Knowledge Graph Convolutional Networks (KGCN) (H. Wang et al. 2019). The solution is an adaptation of the Pytorch implementation in `https://github.com/zzaebok/KGCN-pytorch`. Concerning the adaptations, `KGCN/main_kgcn.ipynb` introduces a pre-processing routines to adapt the KG obtained in Section 3.1 and make it usable with KGCN, as well as testing/evaluation routines to compute recommendation metrics. (LO3)

3.3 Differentially private embeddings

As described in Section 2.4, the application of DP to KG embeddings requires the embeddings to be gradient-separable. Therefore, a differentially private version of the embedding (and thus recommendation) procedure has been implemented for TransE and TransM only. (LO12)

More in detail, achieving DP-SGD requires additional operations with respect to standard SGD,

[3] It should be noted that RDF databases can be considered as a subclass of graph databases and so, in turn, as part of the family of non-relational databases.

namely, gradient clipping[4] and the addition to noise on the gradients. Intuitively, DP aims at limiting the influence that single points (triplets, in this specific case) can have on gradients and, consequently, on the optimisation process. Once a maximum value for gradients has been decided, noise helps obscuring the individual contributions of the points. With DP-SGD, each optimisation step is (ε, δ)-differentially private: because of the composition properties of DP (Dwork, Roth, et al. 2014) the whole procedure is then differentially private.

Concerning implementation aspects, a DP private version of TransE and TransM has been obtained using the private wrapper provided by the Opacus Python library. Opacus includes utilities for privacy accounting, so that the privacy budget ε can be obtained once the models have been trained.

4 Results

This section describes the experimental setup and presents the results of experimentation with the recommendation task.

4.1 Dataset and experimental setup

Experiments were carried out on an 11th Gen Intel i7-1165G7 CPU, with 32GB of RAM available.

Dataset Due to hardware and time constraints, a subset of the relations obtained in Section 3.1 was used for the experiments. Specifically, only the *feedback*, *director*, *writer*, and *music composer* relationships were considered, and only the *user*, *movie*, and *people* entities were considered. This choice significantly reduces the size of the KG and allows for quicker experimentation especially without the need of a GPU. The feedback relationship and, equivalently, the ratings matrix, has been obtained from ml-latest-small, a subset of the MovieLens dataset containing approximately 100 000 ratings for 9000 movies and 600 users. Datasets where split, user-wise, in training, testing, and validation sets following respectively 0.7, 0.2, and 0.1 proportions.

Parameter setting The initialisation of the embeddings follows that of the original TransE and TransM articles (Bordes et al. 2013; Fan et al. 2014). After some experimentation, the dimensionality of the embedding space was set to $k = 50$, the margin in Eq. (2) to $\gamma = 1$, and the learning rate to $\text{lr} = 0.1$. The optimisation was performed using (DP-)SGD with a batch size of 128 for 20 epochs for the standard implementations and for 10 epochs for the DP ones. Concerting DP specific parameters, the maximum gradient norm was set to 1 and $\delta = \frac{\text{batch size}}{\#\text{ training instances}}$. Concerning the KGCN implementation, default values proposed by the existing PyTorch implementation where used. Concerning the WALS, the algorithm was trained for 10 iterations and, in Eq. (1), $\lambda = 0.3$.

4.2 Recommendation

KG-based recommendation can naturally be thought as a link prediction task on the graph (Guo et al. 2020). (LO9)

[4] It should be mentioned that gradient clipping is commonly used in vanilla SGD-like optimisation processes as well, as it is beneficial against exploding and vanishing gradients (Pascanu, Mikolov, and Bengio 2013).

For the TransE and TransM implementation presented in Section 2.2, the task is that of finding triplets that minimise the interaction between entities and relationships described in Eq. (3). Considering all possible combinations of entities and relationships, performance can then be assessed by counting how many of the triplets that show minimum interaction are effectively present in the test set. Performance in this information retrieval task can evaluated by precision measure. In particular, we consider (mean) precision at 10, and so (mean) the number of relevant triplets found among the top 10 retrieved triplets (i.e., the 10 triplets that minimise Eq. (3) for each batch of data considered).

The training procedure can be monitored by means of the loss value in Eq. (2) and by the percentage of training triplets for which a minimisation of the interaction in Eq. (3) has not been achieved yet.

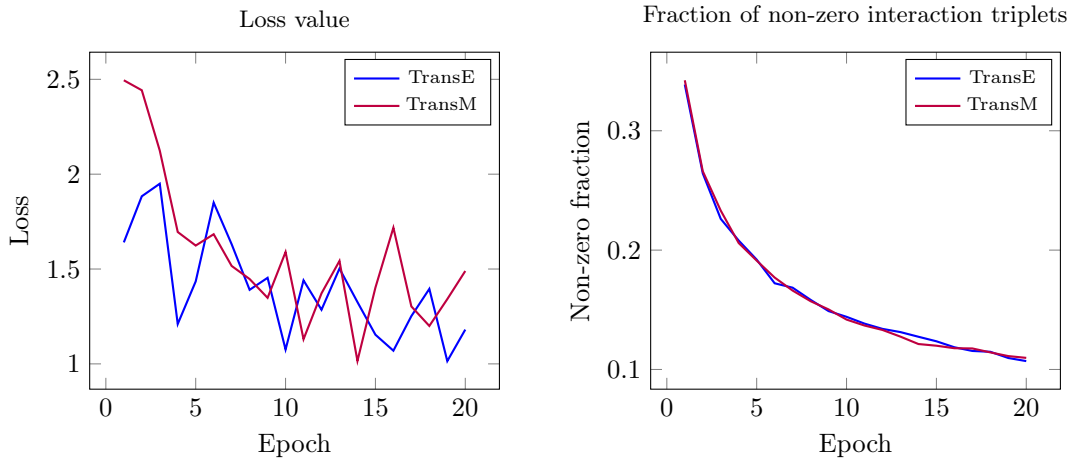

Figure 1: Training loss and fraction of non-zero interaction triplets for TransE and TransM.

It can be observed that, despite a jagged trend for the loss values, the fraction of non-zero interaction triplets gradually approaches a fraction of ≈ 0.1. The optimisation with DP-SGD results in similar trends but with the fraction of non-zero interaction triplets approaching a significantly worse value of ≈ 0.3.

Model	Training time (minutes)	Test time (minutes)	Precision@10
TransE	≈ 1	≈ 0.5	32.2 (40.0)
TransM	≈ 2	≈ 0.5	32.0 (39.6)
DP-TransE	≈ 50	≈ 1	0.8 (0.9)
DP-TransM	≈ 90	≈ 1	0.9 (0.9)
KGCN	≈ 15	≈ 0.5	10.4
WALS	≈ 50	≈ 2	2.7

Table 1: Precision at 10 for the various approaches (values as % numbers). The precision reported in parentheses refers to using a test set that contains the *feedback* relationship only. It should be pointed out that time measures are likely imprecise and are to be taken in terms of order of magnitude.

Table 1 summarises the results of the experiments. TransE and TransM perform quite similarly, suggesting pertinent triplets in more than one third of the top 10 recommendations. Particularly,

the movie recommendation task (denoted in parenthesis in the table), i.e., when predictions are computed only for the *feedback* relationship, reach a precision of $\approx 40\%$.

Performance decreases drastically when DP is introduced to the methods: the models take significantly longer to train (due to the additional procedures necessary to ensure DP and monitor the privacy budget) and perform radically worse, essentially providing no good recommendations.

Despite being optimised for binary classification, and thus not for ranking results, KGCN performs well in the top 10 recommendation task. WALS is bested by the embedding approaches and takes significantly longer to train, with the provided implementation.

5 Discussion

In this section, the further considerations about the results of the experiments, how to improve them, and how to eventually update and deploy the recommender system will be made.

5.1 Performance of the differentially private approach

The unsatisfactory results of the DP approach can be partially justified by means of a more detailed examination of the implemented solution. Specifically, the DP framework adopted considers *all* triplets to be sensitive statements which should be protected. Han et al. 2022 shows that this fully private approach can however lead to significantly worse performance if compared to a more refined approach which consider only a subset of the triplets to be sensitive statements: a fully private implementation can lead to a decrease in precision of $\approx 10\%$. The non-fully private solution has however not been implemented, and thus the impact on the specific dataset considered for the project cannot be assessed. (LO12)

5.2 On model update and scalable reasoning

Once the recommender system has been trained on the KG, it is interesting to consider how could the KG and the system itself be evolved and modified. Firstly, it should be noted that the task at hand, that is, recommendation, can be easily framed as a KG completion task and, more specifically, as a link prediction method. Table 1 shows that the proposed method performs well in this task, adding relevant triplets in more than one third of the test instances. As discussed in Section 2.2, entity (and, similarly, relationship) prediction can be obtained by finding triples that minimise Eq. (3). The project has focused on entity (movie) prediction for the *feedback* relationship, but similar approaches can be easily applied to prediction of other entities or relationships. Pragmatically, a new-found triplet could then easily be added to the text file storing the KG. Assuming new triplets are at some point available, the model itself could be updated by means of embedding techniques that update the embedding using online learning algorithms such as the one proposed in Wu et al. 2022. For what concerns modifications in the underlying schema of the KG, in this specific case that would correspond to modifications of the DBpedia RDF schema: RDF schemas allow modifications including the addition and removal of properties and classes. (LO8)

As far as reasoning at scale is concerned, the recommendation task scales at least linearly with the number of entities in the graph as the energy function for TransE/TransM must be computed for all possible movies in the KG. As the resulting scores must then be sorted in order to provide recommendations, providing recommendations scales at least as $n \log n$. For more considerations (LO6) (LO5)

on scalable reasoning, however, potential architecture needs have to be addressed. Specifically, a KG-based recommender system would likely include KG as part of the middleware of the overall recommendation system (not dissimilarly to the approach followed by this project). The underlying, basic data for a larger scale implementation would need to include access to DBpedia and to user-movie interaction. As mentioned, DBpedia provides endpoints and can be queried with SPARQL, and new triplets can be appended to the KG file. In a real-world deployment, DBpedia could be periodically queried for new movies, and the KG consequently periodically updated. In parallel, user data and ratings could be stored in a (distributed) relational database for quick (parallel) access to information, as the database schema describing users, their properties, and ratings can be rigid and is not expected to change. This database can then be queried using an SQL variant. Together with storing new information, data pre-processing could also be performed offline and periodically, to update the KG and user information (i.e., merging new information with the pre-existing one). To deal with new users or new movies, common cold-start solutions would include, e.g., recommending popular movies to new users. Concerning the learning and analysis at scale, distributed (deep) learning and data processing (such as Spark) frameworks exist, with some approaches combining big data access and (deep) learning in a single framework (Kim et al. 2016). Recent results (Zheng et al. 2020) consider, specifically, training KG embeddings at scale, introducing optimisations to improve data locality and parallelisation. Orchestration systems such as Kubernetes can finally be used to handle deployment, management, and scaling of the final application in automated ways.

5.3 Improvements and application to other domains

One of the drawbacks of embedding based recommender systems is that, while easy to implement, they can result in embeddings which are not well suited for the recommendation task. Indeed, joint-learning methods that jointly learn the embedding and the recommendation module can provide better performance (Guo et al. 2020). Recent developments, moreover, exploit attention mechanism in GNN frameworks to model user preferences.

In terms of possible applications to other domains, it is easy to picture applications to the financial domain. Recommender systems (and the task of link prediction they entail) is in fact immediately applicable to, e.g., the banking sector. A KG-based recommender system could, for instance, be a component in a conversational banking tool. Conversational banking tools allow the user to interact with digital banking platforms in a conversational manner (for instance, via text), but easily become too complex for traditional, rule-based approaches. By means of a KG, instead, user's needs could be modelled and recommendations for, e.g., the best insurance package or investment could be based on the contextual information available to the bank and on previous interactions of other users. Among existing applications of KGs to the financial sector, Ren, Long, and Xu 2019 develops an embedding based recommender system for financial news articles in order to provide each user with relevant articles. In the specific case, recommendation leverage on a KG which contains news and information on users, companies, and industry categories. (LO10)

6 Related work

The Netflix Prize (Bennett, Lanning, et al. 2007), a competition sponsored by Netflix where contestants were asked to design an algorithm to recommend movies to users, ignited interest for recommender systems, in general, and for movie recommendation, in particular. A variety of recommender systems algorithms and techniques has since been developed: Schafer et al. 2007 and Bobadilla et al. 2013 provide a survey of these early research efforts.

More recently, several approaches that employ KGs have been proposed (Grad-Gyenge, Filzmoser, and Werthner 2015; Guo et al. 2020). In particular, some several approaches are embedding-based, and thus focus on learning useful embeddings for movies, users, and their interaction (Palumbo, Rizzo, and Troncy 2017; Y. Zhang et al. 2018). Additionally, some efforts enrich datasets with linked data (Ostuni et al. 2013), or employ Graph Neural Networks (GNNs) to produce the embedding and train the recommender system end-to-end (H. Wang et al. 2019).

7 Conclusions

This project has investigated the task of KG-based movie recommendation. A recommender system based on KG embedding has been implemented from scratch using TransE/TransM and compared with matrix factorisation and GNN approaches, showing good performance and efficiency in training. Additionally, the performance of a differentially private version of the recommender system was also assessed and its poor results discussed. Finally, considerations on how the recommender system could be deployed as a service and kept up to date were presented, describing potential approaches to architecture, scalable reasoning, and application to other domains.

8 References

References

Bobadilla, Jesús et al. (2013). "Recommender systems survey". In: *Knowledge-based systems* 46, pp. 109–132.

Schafer, J Ben et al. (2007). "Collaborative filtering recommender systems". In: *The adaptive web*. Springer, pp. 291–324.

Herlocker, Jonathan L et al. (2004). "Evaluating collaborative filtering recommender systems". In: *ACM Transactions on Information Systems (TOIS)* 22.1, pp. 5–53.

Wang, Quan et al. (2017). "Knowledge graph embedding: A survey of approaches and applications". In: *IEEE Transactions on Knowledge and Data Engineering* 29.12, pp. 2724–2743.

Dwork, Cynthia (2008). "Differential privacy: A survey of results". In: *International conference on theory and applications of models of computation*. Springer, pp. 1–19.

Zhang, Yongfeng et al. (2018). "Learning over knowledge-base embeddings for recommendation". In: *arXiv preprint arXiv:1803.06540*.

Bordes, Antoine et al. (2013). "Translating embeddings for modeling multi-relational data". In: *Advances in neural information processing systems* 26.

Fan, Miao et al. (2014). "Transition-based knowledge graph embedding with relational mapping properties". In: *Proceedings of the 28th Pacific Asia conference on language, information and computing*, pp. 328–337.

Guo, Qingyu et al. (2020). "A survey on knowledge graph-based recommender systems". In: *IEEE Transactions on Knowledge and Data Engineering*.

Wang, Hongwei et al. (2019). "Knowledge graph convolutional networks for recommender systems". In: *The world wide web conference*, pp. 3307–3313.

Zhang, Si et al. (2019). "Graph convolutional networks: a comprehensive review". In: *Computational Social Networks* 6.1, pp. 1–23.

Dwork, Cynthia, Aaron Roth, et al. (2014). "The algorithmic foundations of differential privacy". In: *Foundations and Trends® in Theoretical Computer Science* 9.3–4, pp. 211–407.

Han, Xiaolin et al. (2022). "A framework for differentially-private knowledge graph embeddings". In: *Journal of Web Semantics* 72, p. 100696.

Abadi, Martín et al. (Oct. 2016). "Deep Learning with Differential Privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318.

Auer, Sören et al. (2007). "Dbpedia: A nucleus for a web of open data". In: *The semantic web*. Springer, pp. 722–735.

Ostuni, Vito Claudio et al. (2013). "Top-n recommendations from implicit feedback leveraging linked open data". In: *Proceedings of the 7th ACM conference on Recommender systems*, pp. 85–92.

Palumbo, Enrico, Giuseppe Rizzo, and Raphaël Troncy (2017). "Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation". In: *Proceedings of the eleventh ACM conference on recommender systems*, pp. 32–36.

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). "On the difficulty of training recurrent neural networks". In: *International conference on machine learning*. PMLR, pp. 1310–1318.

Wu, Tianxing et al. (2022). "Efficiently embedding dynamic knowledge graphs". In: *Knowledge-Based Systems*, p. 109124.

Kim, Hanjoo et al. (2016). "Deepspark: A spark-based distributed deep learning framework for commodity clusters". In: *arXiv preprint arXiv:1602.08191*.

Zheng, Da et al. (2020). "Dgl-ke: Training knowledge graph embeddings at scale". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 739–748.

Ren, Jiangtao, Jiawei Long, and Zhikang Xu (2019). "Financial news recommendation based on graph embeddings". In: *Decision Support Systems* 125, p. 113115.

Bennett, James, Stan Lanning, et al. (2007). "The netflix prize". In: *Proceedings of KDD cup and workshop*. Vol. 2007. New York, p. 35.

Grad-Gyenge, László, Peter Filzmoser, and Hannes Werthner (2015). "Recommendations on a knowledge graph". In: *1st International Workshop on Machine Learning Methods for Recommender Systems, MLRec*, pp. 13–20.