POLITECNICO DI TORINO
Repository ISTITUZIONALE

An empirical comparison of knowledge graph embeddings for item recommendation

Original
An empirical comparison of knowledge graph embeddings for item recommendation / Palumbo, E., Rizzo, G., Troncy, R., Baralis, E.M., Osella, M., Ferro, E.G.. - ELETTRONICO. - 2106:(2018). (Extended Semantic Web Conference 2018).

Availability:
This version is available at: 11583/2710124 since: 2018-06-26T14:15:51Z

Publisher:
CEUR

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

25 November 2026

An Empirical Comparison of Knowledge Graph Embeddings for Item Recommendation

Enrico Palumbo[1,2,3], Giuseppe Rizzo[1], Raphaël Troncy[2], Elena Baralis[3], Michele Osella[1] and Enrico Ferro[1]

[1] ISMB, Italy,
{palumbo,giuseppe.rizzo, osella, ferro}@ismb.it
[2] EURECOM, France,
raphael.troncy@eurecom.fr
[3] Politecnico di Torino, Italy,
elena.baralis@polito.it

Abstract. In the past years, knowledge graphs have proven to be beneficial for recommender systems, efficiently addressing paramount issues such as new items and data sparsity. At the same time, several works have recently tackled the problem of knowledge graph completion through machine learning algorithms able to learn knowledge graph embeddings. In this paper, we show that the item recommendation problem can be seen as a specific case of knowledge graph completion problem, where the "feedback" property, which connects users to items that they like, has to be predicted. We empirically compare a set of state-of-the-art knowledge graph embeddings algorithms on the task of item recommendation on the Movielens 1M dataset. The results show that knowledge graph embeddings models outperform traditional collaborative filtering baselines and that TransH obtains the best performance.

Keywords: Knowledge Graphs, Recommender Systems, Embedding

1 Background

Recommender systems are traditionally divided in two families: content-based and collaborative filtering algorithms. Content-based algorithms recommend items similar to the set of items that a user has liked in the past, considering the item content, i.e. its metadata. On the other hand, collaborative filtering algorithms look for users that are similar in terms of item preferences and suggest to a user items that similar users have liked. Recently, a great deal of attention has been given to hybrid systems, which combine content-based filtering and collaborative filtering [1]. Knowledge graphs provide an ideal data structure for such systems, as a consequence of their ability of encompassing heterogeneous information, such as user-item interactions and items' relation with other entities, at the same time. Recommender systems leveraging knowledge graphs have shown to be competitive with state-of-the-art collaborative filtering and to efficiently address issues such as new items and data sparsity [15,10,4,11,12].

CEUR-WS.org/Vol-2106/paper2.pdf

In this paper, we show that, when modelling users and items as entities of a knowledge graph, the item recommendation problem can be seen as a specific case of knowledge graph completion problem, where the "feedback" property has to be predicted. Thus, we compare a set of state-of-the-art knowledge graph completion algorithms based on knowledge graph embeddings (TransE [3], TransH [14], TransR [9]) on the problem of item recommendation. The evaluation on the Movielens 1M dataset shows that: 1) knowledge graph embeddings methods outperform two standard collaborative filtering baselines and the "Most Popular" baseline 2) more flexible models such as TransH and TransR achieve better results with respect to the TransE model.

2 Approach

In this paper, we show that the problem of item recommendation can be interpreted as a knowledge graph completion problem (Fig. 1).
Knowledge Graph: we use the definition of knowledge graph given in [12]. A knowledge graph is defined as a set $K = (E, R, O)$ where E is the set of entities, $R \subset ExΓxE$ is a set of typed relations among entities, and O is an ontology, which defines the set of relation types ('properties') Γ. Entities include users $u \in U \subset E$ and items $i \in I \subset E \setminus U$. An observed positive feedback between a user and an item[4] is described by a special property, which we name 'feedback'. In this work, the ontology O is represented by the DBpedia ontology [2].
Item Recommendation: the problem of item recommendation is that of ranking a set of N candidate items $I_{candidates} \subset I$ according to what a user may like. More formally, the problem consists in defining a ranking function $\rho(u, i)$ that assigns a score to any user-item pair $(u, i) \in UxI_{candidates}$ and then sorting the items according to $\rho(u, i)$:

$$L(u) = \{i_1, i_2, ..., i_N\} \tag{1}$$

where $\rho(u, i) > \rho(u, i + 1)$ for any $i = 1..N - 1$.
Knowledge Graph Embeddings: in order to predict missing relations in a knowledge graph, most algorithms rely on feature learning approaches that are able to map entities and relations into a vector space, generating knowledge graph embeddings. In this work, we compare the following models (known as "translational models"):
-**TransE** [3]: learns representations of entities and relations so that $h + l \approx t$ where $(h, l, t) \in R$ is a triple. The score function for a triple is thus $f(h, l, t) = d(h + l, t)$ where d is the Euclidean distance.
-**TransH** [14]: first extension of TransE, enables entities to have different representations when involved in different relations by projecting entities on a hyperplane identified by the normal vector w_l. The score function becomes: $f(h, l, t) = d(h_\perp + l, t_\perp)$, where $h_\perp = h - w_l^T h w_l$ and $t_\perp = t - w_l^T t w_l$.
-**TransR** [9]: enables entities and relations to be embedded in vector space with

[4] Movie ratings are given by users on a 1-5 scale, we assume $r \geq 4$ to be a positive rating.

different dimensions through a projection matrix M_l associated to any relation l. The score function is: $f(h,l,t) = d(h_l + l, t_l)$ where $h_l = hM_l$ and $t_l = tM_l$.

The core idea of using knowledge graph embeddings for item recommendation is that of using the negative score assigned to a triple $f(u, feedback, i)$ as the ranking function $\rho(u,i)$ (Fig. 2). Thus, the approach can be summarized as:
Data splitting: define the set of users' feedback X as a set of triples $(u, feedback, i)$. We split the set of triples X into a X_{train} and X_{test} so that $X = X_{train} \bigcup X_{test}$.
Training: learn the knowledge graph embeddings from K, which includes all the triples in X_{train}, obtaining vector representations of each $e \in E$ and $r \in R$ (including the 'feedback' property)
Testing: for every $u \in U$, sort every $i \in I_{candidates}$ according to the score $\rho(u,i) = -f(u, feedback, i)$

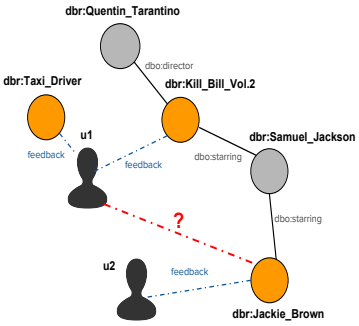

Fig. 1: Recommending items as a knowledge graph completion problem

3 Experimental setup

Knowledge graph construction: the dataset used for the comparison of the knowledge graph embeddings methods is MovieLens 1M[5]. MovieLens 1M [6] is a well known dataset for the evaluation of recommender systems and it contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users. MovieLens 1M items have been mapped to the corresponding DBpedia entities [11] and we leverage these publicly available mappings to create the knowledge graph K using DBpedia data. Since not every item in the Movielens data has a corresponding DBpedia entity, after this mapping we have 948978 ratings, from 6040 users on 3226 items. We split the data into a training X_{train}, validation X_{val} and test set X_{test}, containing, per each user,

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

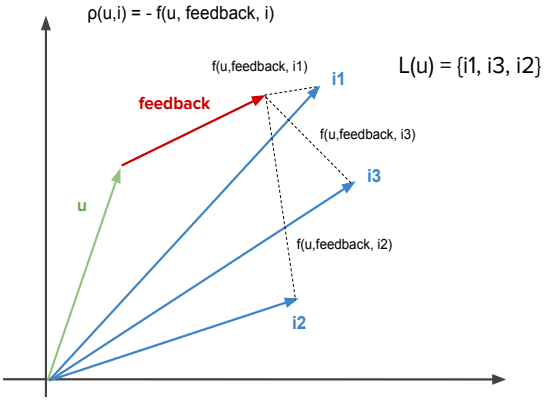

Fig. 2: The ranking function for item recommendation is - f(u, feedback, item), i.e. sorts the items in ascending order according to the distance in vector space.

respectively 70%, 10% and 20% of the ratings. In order to select the most relevant properties for the knowledge graph construction, we count what are the most frequent properties used in DBpedia to describe the items in the Movielens1M dataset and we sort them according to their frequency. We select the first K properties so that the frequency of the K+1 property is less that 50% of the previous one, obtaining: ["dbo:director", "dbo:starring", "dbo:distributor", "dbo:writer","dbo:musicComposer", "dbo:producer", "dbo:cinematography", "dbo:editing"]. We also add "dct:subject" to the set of properties, as it provides an extremely rich categorization of items. For each of these item property p, we include in K all the triples (i, p, e) where $i \in I$ and $e \in E$, e.g. (dbr:Pulp_Fiction, dbo:director, dbr:Quentin_Tarantino). We finally add the 'feedback' property, modeling all movie ratings that are $r \geq 4$ in X_{train} as triples $(u, feedback, i)$.
Evaluation: we use the evaluation protocol known as AllUnratedItems [13], i.e. for each user we select as possible candidate items all the items either in the training or in the test set that he or she has not rated before in the training set. We measure standard information retrieval metrics such as P@5, P@10, Mean Average Precision (MAP), R@5, R@10, NDCG (Normalized Discounted Cumulative Gain), MRR (Mean Reciprocal Rank). As baselines, we use state-of-the-art collaborative filtering algorithms based on Singular Value Decomposition [8], ItemKNN with baselines [7] and the Most Popular Items recommendation strategy, which simply ranks items based on their popularity (i.e. number of positive ratings). All the baselines have been trained on the user ratings contained in X_{train} in the original matrix format and tested on X_{test}. The baselines are implemented using the *surprise* python library[6]. The implementation of the translational based embeddings[7] and the script used to

[6] `http://surprise.readthedocs.io/en/v1.0.2/matrix_factorization.html`
[7] `https://github.com/thunlp/KB2E`

compare them[8] are publicly available on Github. All compared algorithms have been used with their default hyper parameters, as reported in their referenced implementations.

4 Results

The results of the evaluation on the Movielens 1M are reported in Tab. 1. The results show that all knowledge graph embeddings algorithms significantly outperform traditional collaborative filtering baselines such as SVD and ItemKNN. At the same time, we observe that the MostPop baseline, although trivial, is able to achieve very good results, outperforming the TransE method. Note that the MostPop is known to be quite effective on MovieLens due to the power-law distribution of user feedback data, i.e. to the fact that most user ratings tend to be concentrated on few very popular items [5]. On the other hand, the relatively low performance of TransE can be ascribed to the fact that the 'feedback' property is a N-to-N property (a user typically likes N items and an item is liked by N users), which is the typical case where TransE fails to generate good predictions [9,14]. More flexible models such as TransH and TransR are able to effectively model N-to-N properties by allowing entities to have multiple representations and achieve better results in item recommendation. However, the additional flexibility introduced by TransR with respect to TransH in allowing entities and relations to be embedded in different vector spaces does not pay off, but rather leads to a slighty worse performance.

System	P@5	P@10	MAP	R@5	R@10	NDCG	MRR
TransH	**0.196457**	**0.170331**	**0.134170**	**0.076639**	**0.128227**	**0.461370**	**0.396380**
TransR	0.190497	0.165033	0.127401	0.073169	0.121329	0.453900	0.384536
MostPop	0.144603	0.129156	0.092103	0.049231	0.084936	0.406294	0.307453
TransE	0.116656	0.098245	0.071185	0.038067	0.063339	0.379548	0.261642
SVD	0.067815	0.062401	0.042671	0.020201	0.037233	0.328776	0.164112
ItemKNN	0.057483	0.053626	0.040933	0.018734	0.031996	0.324887	0.143604
Random	0.006854	0.006573	0.008482	0.001603	0.003093	0.246370	0.030400

Table 1: Comparison of knowledge graph embeddings and collaborative filtering algorithms sorted by NDCG

5 Conclusions

In this work, we have reported an empirical comparison of knowledge graph embeddings algorithms for item recommendation. First, we have shown that the item recommendation can be interpreted as a knowledge graph completion problem, where a special property called 'feedback', modeling users preferences for

[8] `https://github.com/D2KLab/entity2rec/blob/dev/entity2rec/trans_recommender.py`

items, has to be predicted. Secondly, we have described how to use the predicted score for the 'feedback' property as a ranking function for items. Finally, we have evaluated a set of state-of-the-art knowledge graph embeddings algorithms on the well known Movielens 1M dataset, comparing them and observing that: 1) knowledge graph embeddings algorithms outperform traditional collaborative filtering algorithms for item recommendation 2) flexible models such as TransH and TransR provide better performance with respect to TransE, as a consequence of their ability of modelling N-to-N relations. In a future work, we plan to extend this evaluation to other datasets, to include other existing recommender systems based on knowledge graphs and to take into account specific collaborative filtering issues such as new items and data sparsity.

References

1. Adomavicius, G., Tuzhilin, A.: 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)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: The semantic web, pp. 722–735. Springer (2007)
3. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795 (2013)
4. Catherine, R., Cohen, W.: Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In: Proceedings of the 10th ACM Conference on Recommender Systems. pp. 325–332. ACM (2016)
5. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the fourth ACM conference on Recommender systems. pp. 39–46. ACM (2010)
6. Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5(4), 19 (2016)
7. Koren, Y.: Factor in the neighbors: Scalable and accurate collaborative filtering. ACM Transactions on Knowledge Discovery from Data (TKDD) 4(1), 1 (2010)
8. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8) (2009)
9. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI. vol. 15, pp. 2181–2187 (2015)
10. Noia, T.D., Ostuni, V.C., Tomeo, P., Sciascio, E.D.: Sprank: Semantic path-based ranking for top-n recommendations using linked open data. ACM Transactions on Intelligent Systems and Technology (TIST) 8(1), 9 (2016)
11. Ostuni, V.C., Di Noia, T., Di Sciascio, E., Mirizzi, R.: Top-n recommendations from implicit feedback leveraging linked open data. In: Proceedings of the 7th ACM conference on Recommender systems. pp. 85–92. ACM (2013)
12. Palumbo, E., Rizzo, G., Troncy, R.: 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. ACM (2017)
13. Steck, H.: Evaluation of recommendations: rating-prediction and ranking. In: Proceedings of the 7th ACM conference on Recommender systems. pp. 213–220. ACM (2013)

14. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI. vol. 14, pp. 1112–1119 (2014)
15. Yu, X., Ren, X., Sun, Y., Gu, Q., Sturt, B., Khandelwal, U., Norick, B., Han, J.: Personalized entity recommendation: A heterogeneous information network approach. In: Proceedings of the 7th ACM international conference on Web search and data mining. pp. 283–292. ACM (2014)