Knowledge Graph Embeddings with node2vec 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, Turin, Italy
{palumbo,giuseppe.rizzo,osella,ferro}@ismb.it
[2] EURECOM, Biot, France
raphael.troncy@eurecom.fr
[3] Politecnico di Torino, Turin, 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. Graph embeddings algorithms have shown to be able to automatically learn high quality feature vectors from graph structures, enabling vector-based measures of node relatedness. In this paper, we show how node2vec can be used to generate item recommendations by learning knowledge graph embeddings. We apply node2vec on a knowledge graph built from the MovieLens 1M dataset and DBpedia and use the node relatedness to generate item recommendations. The results show that node2vec consistently outperforms a set of collaborative filtering baselines on an array of relevant metrics.

Keywords: Knowledge graphs embeddings · Recommender systems node2vec

1 Background

In the past few years, recommender systems leveraging knowledge graphs have proven to be competitive with state-of-the-art collaborative filtering systems and to efficiently address issues such as new items and data sparsity [1,7–10,12]. node2vec has shown to be able to effectively learn features from graph structures, outperforming existing systems in node classification and link prediction tasks [3]. In this paper, we show that node2vec can be effectively used to learn knowledge graph embeddings to perform item recommendation. node2vec is applied on a knowledge graph including user feedback on items, modelled by the special relation 'feedback', and item relations to other entities. Then, recommendations are generated using the relatedness between users and items in the vector space. The evaluation on the Movielens dataset shows that: (1) node2vec with default hyper-parameters outperforms collaborative filtering baselines on all metrics and the MostPop algorithm on most metrics (2) node2vec with optimized hyper-parameters significantly outperforms all baselines under consideration.

2 Approach

Item Recommendation: given a set of items I and a set of users U, 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 $(u, i) \in U \times I_{candidates}$ and then sorting the items according to $\rho(u, i)$:

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

where $\rho(u, i_j) > \rho(u, i_{j+1})$ for any $j = 1..N - 1$.

node2vec [3] learns representations of nodes in a graph through the application of the word2vec model on sequences of nodes sampled through random walks (Fig. 1). The innovation brought by node2vec is the definition of a random walk exploration that is flexible and adaptable to the diversity of connectivity patterns that a network may present. Given a knowledge graph K encompassing users U, items I (the object of the recommendations, e.g. a movie) and other entities E (objects connected to items, e.g. the director of a movie), node2vec generates vector representations of the users x_u and of the items x_i (and of other entities x_e). Thus, we propose to use as a ranking function the relatedness between the user and the item vectors: $\rho(u, i) = d(x_u, x_i)$ where d is the cosine similarity in this work.

Knowledge Graph Construction: the dataset used for the evaluation is MovieLens 1M[1] [4]. We used the publicly available mappings from MovieLens 1M items to the corresponding DBpedia entities [8] to create the knowledge graph K using DBpedia data. We split the data into training X_{train}, validation X_{val} and test set X_{test}, containing respectively 70%, 10% and 20% of the ratings for each user. We selected a set of properties based on their frequency of occurrence[2]: ["dbo:director", "dbo:starring", "dbo:distributor", "dbo:writer","dbo:musicComposer", "dbo:producer", "dbo:cinematography", "dbo:editing"]. We add "dct:subject" to this set of properties, as it provides an extremely rich categorization of items. For each 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 [11] and 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 collaborative filtering algorithms based on Singular Value Decomposition [6], ItemKNN with baselines [5] and the MostPop item recommendation strategy, which ranks items

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

[2] We sorted the properties used in DBpedia to describe the Movielens1M items according to their frequency and selected the first K properties so that the frequency of the K + 1 property was less that 50% of the previous one.

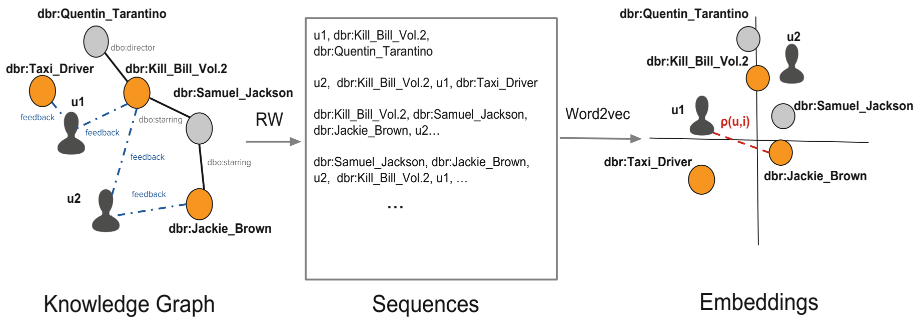

Fig. 1. Node2vec for item recommendation using the knowledge graph. Users are represented in black, items in orange and entities in grey. node2vec learns knowledge graph embeddings by sampling sequences of nodes through random walks and then applying the word2vec model on the sequences. The ranking function for item recommendation is then given by the node relatedness in the vector space.

based on their popularity (i.e. total number of positive ratings). The baselines are implemented using the *surprise* python library[3].

3 Results

The results of the evaluation are reported in Table 1. In "node2vec (default)" the hyper-parameters have been set to their default value as reported in the original paper [3] and in the reference Python implementation available on Github[4] ($p = 1, q = 1, num\_walks = 10, walk\_length = 80, window\_size = 10, iter = 1, dimensions = 128$). We observe that "node2vec (default)" outperforms SVD and ItemKNN on all metrics, but that the MostPop approach performs slightly better on the P@5 and P@10 and on MRR. Note that MostPop, although trivial, is known to be quite effective on the Movielens 1M dataset as a consequence of the strong concentration of item feedback on a small number of highly popular items [2]. In "node2vec (opt)" we have optimized the hyper-parameters by a combination of grid-search and manual search over the validation set, exploring the ranges: $p \in \{0.25, 1, 4\}$, $q \in \{0.25, 1, 4\}$, $dimensions \in \{200, 500\}$, $walk\_length \in \{10, 20, 30, 50, 100\}$, $window\_size \in \{10, 20, 30\}$, $num\_walks \in \{10, 50\}$. We found the configuration ($p = 4, q = 1, num\_walks = 50, walk\_length = 100, window\_size = 30, iter = 5, dimensions = 200$) to be optimal on the validation set in the explored range. We observed that the number of walks per node, the walk length, i.e. the maximum length of random walk, and the context size are particularly significant to improve the performance. However, the hyper-parameters optimization is a time consuming endeavor, as it requires running the whole evaluation pipelines with multiple configurations. Thus, in a future work, we will extend the evaluation to other datasets and investigate the relation between hyper-parameters and the graph structure, with the aim of elaborating some indications to guide the hyper-parameter search process.

[3] http://surprise.readthedocs.io/en/v1.0.2/matrix_factorization.html.
[4] https://github.com/aditya-grover/node2vec.

Table 1. Results on the MovieLens 1M dataset sorted by NDCG

System	P@5	P@10	MAP	R@5	R@10	NDCG	MRR
node2vec (opt)	**0.224**	**0.196**	**0.153**	**0.092**	**0.155**	**0.482**	**0.441**
node2vec (default)	0.120	0.129	0.117	0.067	0.122	0.438	0.256
MostPop	0.145	0.129	0.092	0.049	0.085	0.406	0.307
SVD	0.068	0.062	0.043	0.020	0.037	0.329	0.164
ItemKNN	0.057	0.054	0.041	0.019	0.032	0.325	0.143
Random	0.007	0.007	0.008	0.002	0.003	0.246	0.030

References

1. 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)
2. 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)
3. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864. ACM (2016)
4. Harper, F.M., Konstan, J.A.: The movielens datasets: history and context. ACM Trans. Interact. Intell. Syst. (TiiS) **5**(4), 19 (2016)
5. Koren, Y.: Factor in the neighbors: scalable and accurate collaborative filtering. ACM Trans. Knowl. Discov. Data (TKDD) **4**(1), 1 (2010)
6. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer **42**(8), 30–37 (2009)
7. 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 Trans. Intell. Syst. Technol. (TIST) **8**(1), 9 (2016)
8. 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)
9. 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)
10. Rosati, J., Ristoski, P., Di Noia, T., Leone, R.d., Paulheim, H.: RDF graph embeddings for content-based recommender systems. In: CEUR Workshop Proceedings, vol. 1673, pp. 23–30. RWTH (2016)
11. Steck, H.: Evaluation of recommendations: rating-prediction and ranking. In: Proceedings of the 7th ACM Conference on Recommender systems, pp. 213–220. ACM (2013)
12. Yu, X., et al.: 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)