Explainable Entity-based Recommendations with Knowledge Graphs

Rose Catherine
School of Computer Science
Carnegie Mellon University, USA
rosecatherinek@cs.cmu.edu

Kathryn Mazaitis
School of Computer Science
Carnegie Mellon University, USA
krivard@cs.cmu.edu

Maxine Eskenazi
School of Computer Science
Carnegie Mellon University, USA
max@cs.cmu.edu

William Cohen
School of Computer Science
Carnegie Mellon University, USA
wcohen@cs.cmu.edu

ABSTRACT

Explainable recommendation is an important task. Many methods have been proposed which generate explanations from the content and reviews written for items. When review text is unavailable, generating explanations is still a hard problem. In this paper, we illustrate how explanations can be generated in such a scenario by leveraging external knowledge in the form of knowledge graphs. Our method jointly ranks items and knowledge graph entities using a Personalized PageRank procedure to produce recommendations together with their explanations.

1 INTRODUCTION

Improving the accuracy of predictions in recommender systems is an important research topic. An equally important task is explaining the predictions to the user. Providing an explanation has been shown to build user's trust in the recommender system [9].

The focus of this paper is a system to generate explanations for Knowledge Graph (KG) -based recommendation. Users and items are typically associated with factual data, referred to as *content*. For users, the content may include demographics and other profile data. For items such as movies, it might include the actors, directors, genre, and the like. The KG encodes the interconnections between such facts, and leveraging these links has been shown to improve recommender performance [2, 3, 13].

Although a number of explanation schemes have been proposed in the past (Section 2), there has been no work which produces explanations for KG-based recommenders. In this paper, we present a method to jointly rank items and entities in the KG such that the entities can serve as an explanation for the recommendation.

Our technique can be run without training, thereby allowing faster deployment in new domains. Once enough data has been collected, it can then be trained to yield better performance. The proposed method can also be used in a dialog setting, where a user interacts with the system to refine its suggestions.

2 RELATED WORK

Generating explanations for recommendations has been an active area of research for more than a decade. [4] was an early work that assessed different ways of explaining recommendations in a collaborative filtering (CF) -based recommender system. In content-based recommenders, the explanations revolve around the content or profile of the user and the item. The system of [1] simply displayed keyword matches between the user's profile and the books being recommended. Similarly, [11] proposed a method called 'Tagsplanations', which showed the degree to which a tag is relevant to the item, and the sentiment of the user towards the tag.

With the advent of social networks, explanations that leverage social connections have also gained attention. For example, [10] produced explanations that showed whether a good friend of the user has liked something, where friendship strength was computed from their interactions on Facebook.

More recent research has focused on providing explanations that are extracted from user written reviews for the items. [14] extracted phrases and sentiments expressed in the reviews and used them to generate explanations. [5] uses topics learned from the reviews as aspects of the item, and uses the topic distribution in the reviews to find useful or representative reviews.

Knowledge Graphs have been shown to improve the performance of recommender systems in the past. [13] proposed a meta-path based method that learned paths consisting of node types in a graph. Similarly, [7] used paths to find the top-N recommendations in a learning-to-rank framework. A few methods such as [3, 6] rank items using Personalized PageRank. In these methods, the entities present in the text of an item are first mapped to entities in a knowledge graph. [2] proposed probabilistic logic programming models for recommendation on knowledge graphs. None of the above KB-based recommenders attempted to generate explanations.

3 EXPLANATION METHOD

In this section, we propose our method, which builds on the work of [2] by using ProPPR [12] for learning to recommend. ProPPR (**Pro**gramming with **P**ersonalized **P**age **R**ank) is a first order logic system. It takes as input a set of rules and a database of facts, and uses these to generate an approximate local grounding of each query in a small graph. Candidate answers to the query are the nodes in the graph that satisfy the rules. The candidates are then ranked by running a Personalized PageRank algorithm on the graph.

Our technique proceeds in two main steps. First, it uses ProPPR to jointly rank items and entities for a user. Second, it consolidates the results into recommendations and explanations.

To use ProPPR to rank items and entities, we first define a notion of similarity between nodes in the graph, using the same similarity rules as [2] (Figure 1). This simple rule states that two entities `X` and `E` are similar if they are the same (Rule 1), or if there is a link in the graph connecting `X` to another entity `Z`, which is similar to `E` (Rule 2). Note that this definition of similarity is recursive.

RecSys '17 Poster Proceedings, Como, Italy

$$\texttt{sim(X, X)} \leftarrow \texttt{true}. \quad (1)$$

$$\texttt{sim(X, E)} \leftarrow \texttt{link(X, Z), sim(Z, E)}. \quad (2)$$

Figure 1: Similarity in a graph

Next, the model has two sets of rules for ranking: one set for joint ranking of movies that the user would like, together with the most likely reason (Figure 2), and a similar set for movies that the user would not like. In Figure 2, Rule 3 states that a user `U` will like an entity `E` and a movie `M` if the user likes the entity, and the entity is related (`sim`) to the movie. The clause `isMovie` ensures that the variable `M` is bound to a movie, since `sim` admits all types of entities. Rule 3 invokes the predicate `likes(U,E)`, which holds for an entity `E` if the user has explicitly stated that they like it (Rule 4), or if they have provided positive feedback (e.g. clicked, thumbs up, high star rating) for a movie `M` containing (via `link(M,E)`) the entity (Rule 5). The method for finding movies and entities that the user will dislike is similar to the above, except 'like' is replaced with 'dislike'.

$$\texttt{willLike(U, E, M)} \leftarrow \texttt{likes(U, E), sim(E, M), isMovie(M)}. \quad (3)$$

$$\texttt{likes(U, E)} \leftarrow \texttt{likesEntity(U, E)}. \quad (4)$$

$$\texttt{likes(U, E)} \leftarrow \texttt{likesMovie(U, M), link(M, E)}. \quad (5)$$

Figure 2: Predicting likes

To jointly rank the items and entities, we use ProPPR to query the `willLike(U,E,M)` predicate with the user specified and the other two variables free. Then, the ProPPR engine will ground the query into a proof graph by replacing each variable recursively with literals that satisfy the rules from the KG [2, 12]. A sample grounding when queried for a user `alice` who likes `tom_hanks` and the movie `da_vinci_code` is shown in Figure 3.

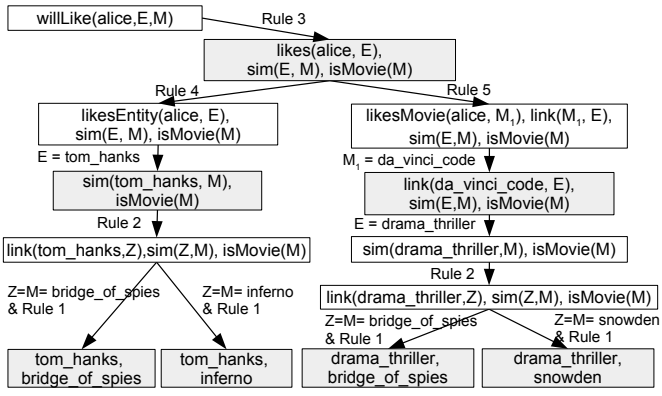

Figure 3: Sample grounding for predicting likes

After constructing the proof graph, ProPPR runs a Personalized PageRank algorithm with `willLike(alice, E, M)` as the start node. In this simple example, we will let the scores for `(tom_hanks, bridge_of_spies)`, `(tom_hanks, inferno)`, `(drama_thriller, bridge_of_spies)`, and `(drama_thriller, snowden)`, be 0.4, 0.4, 0.3 and 0.3 respectively.

Now, let us suppose that `alice` has also specified that she dislikes `crime` movies. If we follow the grounding procedure for dislikes and rank the answers, we may obtain `(crime, inferno)` with score 0.2. Our system then proceeds to consolidate the recommendations and the explanations by grouping by movie names, adding together their 'like' scores and deducting their 'dislike' scores. For each movie, the entities can be ranked according to their joint score. The end result is a list of reasons which can be shown to the user:

(1) `bridge_of_spies`, score = 0.4 + 0.3 = 0.7, reasons = { tom_hanks, drama_thriller }
(2) `snowden`, score = 0.3, reasons = { drama_thriller }
(3) `inferno`, score = 0.4 - 0.2 = 0.2, reasons = { tom_hanks, (-ve) crime }

4 REAL WORLD DEPLOYMENT

The proposed method is presently used as the backend of a personal agent running on mobile devices for recommending movies [8] undergoing Beta testing. The knowledge graph for recommendations is constructed from the weekly dump files released by imdb.com. The personal agent uses a dialog model of interaction with the user. In this setting, users are actively involved in refining the recommendations depending on what their mood might be. For example, for a fun night out with friends, a user may want to watch an action movie, whereas when spending time with her significant other, the same user may be in the mood for a romantic comedy.

5 CONCLUSIONS

Knowledge graphs have been shown to improve recommender system accuracy in the past. However, generating explanations to help users make an informed choice in KG-based systems has not been attempted before. In this paper, we proposed a method to produce a ranked list of entities as explanations by jointly ranking them with the corresponding movies.

ACKNOWLEDGMENTS

This research was supported in part by Yahoo! through the CMU-Yahoo InMind project.

REFERENCES

[1] Mustafa Bilgic and Raymond J. Mooney. 2005. Explaining Recommendations: Satisfaction vs. Promotion. In *Beyond Personalization Workshop*.
[2] R. Catherine and W. Cohen. 2016. Personalized Recommendations Using Knowledge Graphs: A Probabilistic Logic Programming Approach. In *Proc. RecSys '16*. 325–332.
[3] S. Chaudhari, A. Azaria, and T. Mitchell. An Entity Graph Based Recommender System. In *RecSys '16 Posters*.
[4] J. Herlocker, J. Konstan, and J. Riedl. 2000. Explaining collaborative filtering recommendations.. In *CSCW*. 241–250.
[5] J. McAuley and J. Leskovec. Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In *RecSys '13*. 165–172.
[6] C. Musto, P. Basile, M. de Gemmis, P. Lops, G. Semeraro, and S. Rutigliano. Automatic Selection of Linked Open Data features in Graph-based Recommender Systems. In *CBRecSys 2015*.
[7] V. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-N Recommendations from Implicit Feedback Leveraging Linked Open Data. In *RecSys '13*. 85–92.
[8] F. Pecune, T. Baumann, Y. Matsuyama, O. Romero, S. Akoju, Y. Du, R. Catherine, J. Cassell, M. Eskenazi, A. Black, and W. Cohen. InMind Movie Agent - A Platform for Research *(In Preparation)*.
[9] P. Pu and L. Chen. Trust Building with Explanation Interfaces. In *IUI '06*. 93–100.
[10] A. Sharma and D. Cosley. 2013. Do Social Explanations Work?: Studying and Modeling the Effects of Social Explanations in Recommender Systems. In *WWW '13*. 1133–1144.
[11] J. Vig, S. Sen, and J. Riedl. Tagsplanations: Explaining Recommendations Using Tags. In *IUI '09*. 47–56.
[12] W. Wang, K. Mazaitis, and W. Cohen. Programming with Personalized Pagerank: A Locally Groundable First-order Probabilistic Logic. In *Proc. CIKM '13*.
[13] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized Entity Recommendation: A Heterogeneous Information Network Approach. In *WSDM '14*. 283–292.
[14] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma. Explicit Factor Models for Explainable Recommendation Based on Phrase-level Sentiment Analysis. In *SIGIR '14*. 83–92.