An Entity Graph Based Recommender System

Sneha Chaudhari
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA
E-mail: sschaudh@andrew.cmu.edu

Amos Azaria
Computer Science Department, Ariel University, Israel
E-mail: amos.azaria@ariel.ac.il

Tom Mitchell
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA
E-mail: tom.mitchell@cs.cmu.edu

Recommender Systems have become increasingly important and are applied in an increasing number of domains. While common collaborative methods measure similarity between different users, common content based methods measure similarity between different content. We propose a privacy aware recommender system that exploits relations present between entities appearing in content from user's history and entities appearing in candidate content. In order to identify such relations, we use the knowledge graph of NELL, which encodes entities and their relations. We present a novel normalized version of Personalized PageRank, to rank candidate content. We test our approach on the movie recommendation domain and show that the proposed method outperforms other baseline methods, including the standard Personalized PageRank. We intend to deploy our recommender system as a news recommendation app for mobile devices.

Keywords: Recommender Systems, Knowledge-Graphs, PageRank

1. Introduction

With the diverse and explosive growth of web information, organization and utilization of the information effectively and efficiently has become more and more important. Information filtering systems which present only relevant content to the users, have become indispensable. Recommendation Systems [7] are such information filtering systems that attempt to narrow down the information relevant to the users based on their expressed preferences, past behavior, or other data and generate meaningful recommendations to users for items or products that might interest them. Innumerable different kinds of recommendations are made on the web every day, for example, suggestions for books on Amazon [28], or movies on Netflix [6], are real world examples of the operation of industry-strength recommendation systems. They have also been widely adopted in online applications by E-commerce websites to suggest products, services, and contents to potential users.

A major challenge for recommender systems is the cold start problem [29,15]. This problem occurs any-time a new user joins the system and any-time a new product or content is added. In many websites the majority of users are one-time users and thus must be considered as new, and in other types of websites the turnover of the content is so rapid that not much information is gathered on each article. This problem is also present if a user does not log-in to the system and rejects web-cookies.

Privacy aware users are concerned about systems that collect all their purchase history and make recommendations to other users based on their own purchases[2]. Due to the increase in privacy awareness, many major websites (e.g. Amazon and Google) allow the user to disallow the system to record their history. However, as a result, the recommendations that are given to the user are completely irrelevant. We believe that some privacy aware users would be willing to explicitly state their interests (in a general non-domain dependent manner), if this would entail that the recommendations would become more relevant to them.

Suppose a customer at a movie rental store discusses his preferences with a saleswomen. The customer is not likely to provide a long list of movies that he had enjoyed watching in the past. Nor will this customer list any friends that may have similar preference. What people do in such cases, is providing a list of few *entities* that they are interested in (e.g. “Tom Cruise”, “Science-Fiction” and “Mars”) and perhaps also list a movie or two that they enjoyed watching in the past (e.g. “The Matrix”). This small set of interests alone will allow the saleswomen to recommend a movie to the customer. The recommended movie may not have any of the specific words that the user has mentioned, in our example, suppose the saleswoman recommends the movie “Avatar”. This movie may not necessarily be classified as a Science-Fiction movie (but as a fantasy movie), and by no means does it take place on Mars, but on a fictional planet moon in Alpha Centauri. Furthermore, humans may recommend content to other people based on interests obtained from domains different than the current domain in question. For example, if a user is interested in space, she is most likely interested in movies, books, news articles about space as well as items such as binoculars and telescopes or space related posters.

In this paper we present the Relation of Entities Recommendation Agent (RERA), a new content-based system that takes into account relations that may exist between entities that appear in the users past consumed content and entities that appear in the suggested content. For example, consider the following scenario. Suppose some user has read articles about a football team “Real Madrid” in the past. The recommender system needs to decide whether to recommend a new news article that has mentions of “James Rodriguez” in it. A relation between “James Rodriguez” and “Real Madrid” (“James Rodriguez” plays for the team “Real Madrid”), can hint that this new news article is highly relevant and present it to the user.

RERA uses a knowledge graph, in which the nodes are real world entities (e.g. “James Rodriguez”) and the edges are the relations between them (e.g. PlaysOnTeam(James Rodriguez, Real Madrid)). RERA extracts the entities that appear in the content consumed by the user (or those that appear in the description of the items purchased by the user); this first set of entities is assumed to be the user’s interests. Once RERA is required to recommend new items, it extracts all entities that appear in each of the proposed new items (the entire list of available items). RERA then uses relations that exist in the knowledge graph between the entities, in each of these new items, and the entities in the user’s interests in order to rank these new items. Since RERA is a content-based approach and not collaborative-based it does not require any information on preferences of other users, that is, users other than the user it is currently serving (other graph based approaches such as [22], require preferences from all users in order to be able to serve a single user).

Relying on the relations between entities in a knowledge graph to provide personalized recommendations, is of great importance, as it allows recommendations to be domain independent. That is, RERA can provide recommendations in one domain, based on a single user’s interest in another domain. Using the relations between entities further implies that RERA does not require the mention of the exact same entities (or words) to reveal related content. RERA also reduces privacy concerns as every user receives recommendation solely based on their own preferences, and one user’s history does not affect at all, in any way, the recommendations that will be viewed by another user. This approach also reduces the problem of cold-start for new or rare content, which hasn’t been consumed by many users.

We have enriched the Never-Ending Language Learner (NELL) [20] with Subject-Verb-Object (SVO) triples from the ClueWeb 2009 dataset [8] to serve as RERA’s knowledge graph. In order to rank the suggested content, RERA uses a novel raking method which takes into account both the PageRank of each entity and its *Personalized* PageRank (PPR) [10]. We test RERA in a movie domain (using the MovieLens data-set, combined with movie synopsises crawled from IMDB) and show that RERA, using our novel version of PPR, outperforms PPR and other baseline methods. We further show that RERA benefits from knowing that a user enjoyed as little as 5 movies, without relying on any information related to those movies or other ones from other users.

2. Related Work

At the high level, recommender systems can be classified into two categories, (i) collaborative approaches [32,1,9], and (ii) content-based methods [17,19]. Both methods build upon user's history, such as an indication of whether each user has bought or viewed an item and user ratings to recommend either a product or some content. Collaborative methods [32,1,9] make recommendations to each user based on the similarity between different users. For example, if two users have liked a few identical movies in the past, and one has liked another movie that the second user hasn't watched, a collaborative-based recommender system is likely to recommend that new movie to the second user. Collaborative-based systems require a large set of data from many users to provide effective recommendations.

Content-based methods [17,19] on the other hand, do not require a large data-set from many users, and use the similarity between the products (or content) that the user has previously purchased (or viewed) and the new proposed items (or content). For example, if the user has previously liked many action movies, a content-based recommender system is likely to recommend another action movie.

The commonly used bag-of-words approach [34,21] limits the vocabulary to the words present in the data and relies on an exact match of the words for computing similarity. To overcome this limitation semantic based recommendations have emerged [31,11,3]. These methods address the *polysemy* (the presence of multiple meanings for one word) and the *synonymy* (multiple words with the same meaning) problems present in the bag-of-words-approach. These methods try accessing the meaning of the word rather than simply the exact word. However, previous semantic based recommendation methods do not use the *connections* that may exist between different entities. For example, if a semantic based recommender system learns that a user is interested in movies about "cats", it may recommend movies on "felines", but not on "dogs", despite being very closely related (as both are very often home pets).

There have been many graph-based approaches for recommender systems [33,12,18,14,22,23,13,16, 25,30]. However, in all these approaches the items (or content) to recommend appear as nodes in the graph. We therefore term these approaches *item*-graph-based recommender systems. This is to differentiate between our *entity*-graph-based approach in which the nodes in the graph consist of entities in the knowledge base. [13, 16] are collaborative-based methods, in which, nodes represent items. The edges on the other hand represent similarity between items. These similarities are learned from the attitude different users have to different items. For example, if all users that liked item a, also liked item b, it is likely that there will be an edge between a and b. Furthermore, these edges are weighted in proportion to the similarity between the two items. In both [13] and [16], every user has her own graph, depending on the items the user liked (or items liked by users similar to the current user). Once again, this group of item-graph-based recommender systems, require data from many users in order to provide a recommendation to a specific user. [33,12,18,14] use a bipartite graph in which one set of nodes represent items, while the second set of nodes represents users. Edges represent users liking or buying specific items. For example, if user A bought items a, b and c, there exist corresponding edges in the graph from A to a, b and c. Different works use this graph in various methods, with a goal to find items on the graph that should be recommended to the user. Some works [22,23] use a third type of nodes such as concepts, which allow additional connections between some items to others not necessarily via nodes that represent users. All these item-graph-based approaches are in-fact collaborative based, and thus require data from many users in order to perform well. These methods are also not suitable for users who are highly privacy aware, as information from some users must be used in order to recommend items to other users. As mentioned above, unlike all these item-graph-based recommender systems, in our work, nodes represent entities (e.g. "Mars"), not items (e.g. "Avatar") nor users. Edges in our work represent relations (e.g. "Mars" and "planet" have an edge connecting the two, since they have an "is-a" relationship). Another difference from other graph-based recommender systems is the distance algorithm. While other graph-based recommender systems, rely mostly on the PageRank algorithm [22,24], we present novel method, which normalizes a personalized PageRank by the PageRank, to determine whether connections between different nodes are valuable. We show that our normalized method outperforms the PageRank and personalized PageRank methods.

3. The Knowledge-Base Graph

NELL is a computer system that reads the web and accumulates knowledge over time. The aim of NELL is to extract structured information from the unstructured data present on the web. Further, it learns from this extracted information to improve its reading competence. This structured information in the form of entities, categories and their relations is then converted into structured facts and beliefs as NELL continues to learn. To accomplish this, NELL adopts a semi-supervised learning approach. The inputs to NELL include (1) an initial ontology defining hundreds of categories and (2) 10 to 15 seed examples of each category and relation. NELL then extracts more instances of these categories and relations from millions of web pages. Moreover, it learns to find these categories and relations more accurately over time using previously learned knowledge to improve the subsequent learning.

The resulting knowledge-base of NELL contains the information present on the web in the form of entities, their categories and their relations, which are basically the facts/beliefs NELL has learned from the web. We primarily use this knowledge base of entities, categories and relations for recommendation. While the relations in NELL are directed, every relation has an inverse relation and therefore the graph can be viewed as an undirected graph. For example, if there exists a relationship of “AnimalEatsPlant” between “Cow” and “Grass”, there exists the inverse relationship of “PlantConsumedByAnimal”, between “Grass” and “Cow”. The NELL knowledge graph has approximately 1.5 million entities, and 4.5 million relations among them accounting for an average degree of 3 for each entity.

In order to increase the connectivity level of the knowledge graph, we enrich NELL’s knowledge graph with additional edges by incorporating Subject Verb Object (SVO) triples collected from dependency parses of 500 million web pages from ClueWeb 2009 [8]. For each of these SVO triples, if we find a matching entity in NELL for both the subject and the object, then we add a corresponding edge between these two entities in the NELL knowledge graph. This process was very successful as it increased the number of relations to 8 million (while still having 1.5 million entities), resulting in the average degree increasing to 5.3 for each entity. Figure 1 illustrates the process of adding edges to the NELL graph using the SVO triples.

4. Relation of Entities Recommendation Agent (RERA)

RERA receives as input a set of documents (which could be movie synopsis, new articles or item descriptions) that the user either liked, in the past, or has shown interest in. RERA assumes that each item is basically a set (or bag) of NELL entities mentioned in the article. At first, RERA extracts noun phrases

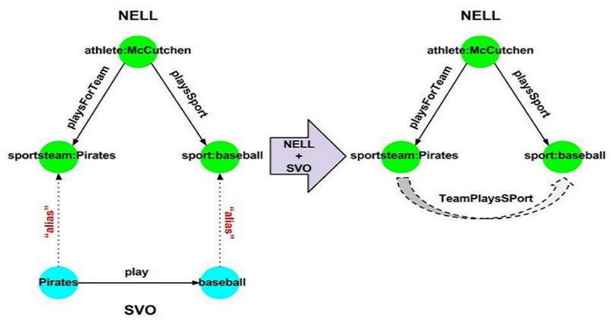

Fig. 1. Example of adding SVO triples to NELL

from these item descriptions using Stanford CoreNLP. Then each of these noun phrases are given as input to NELL’s knowledge on demand API to extract the corresponding NELL entities. This set is assumed to be the entities that the user is interested in, or the set of the user’s interests. In order to provide recommendations, RERA extracts NELL entities from the proposed content using the same method. The basic intuition behind RERA is that if the NELL entities present in the document are well connected with the NELL entities of the user interest, then the content should be highly relevant.

We now formally define the entity graph based recommendation problem. The input consists of the knowledge base, which is a directed graph $G = (V,E)$, where V is the vertex set, $V = \{v_1;v_2;...;v_n\}$ and E is the set of all edges, $E = \{(v_i,v_j)$ if there exists an edge from v_i to $v_j\}$. The vertex set is the NELL entities or concepts and the edges are the relations between these entities. The input also includes a set of entities (or vertices) extracted from user interests $U = \{u_1,u_2,...u_m\}$, where $u \in V$; and a set of documents $\mathcal{D} = \{D_1,D_2,...D_k\}$, where each document is referred to as a set of entities $(D = d_1,d_2,...)$ where $d_i \in V$. Given this information, the entity graph based recommendation problem is to provide a ranking over the set of documents.

Our proposed method follows a two step approach. In the first stage RERA finds the entities that directly appear in the set of the user’s interests (e.g. directly appear in synopsis of movies liked by the user). In the second step, RERA estimates how well connected are the user entities and the entities present in each of the candidates.

For a given document D, in order to determine whether it is well connected to the user interests U we consider three methods. The first method we consider is the PageRank algorithm [24]. This method computes a score for each node in the graph, which denotes a rough estimate of how important the node is, or equivalently, what is the probability of visiting that node in a random walk. This method recommends the documents with the highest average PageRank score (the average is computed over all entities in every document). We use W to denote the weighted transition matrix of graph G where transition from i to j is given by $W_ij = 1/degree(v_i)$. u is a normalized teleportation vector where $|u| = |V|$ and $||u||_1 = 1$ (i.e. $\sum_i u_i = 1$). The page rank algorithm returns as output a scoring vector r over all nodes, satisfying the following equation:

$$r^{PR} = \alpha u + (1-\alpha)W r^{PR} \tag{1}$$

of a random walk on G arriving at node v, with teleportation probability α at every step to a node with distribution u. In PageRank, u is simply a uniform vector. The motivation behind PageRank comes from a web surfer who follows links on the web and with probability α jumps to a random page. We use $PR(v)$ to denote the PR score that v obtained in r^{PR}. The PR score of a given document D is obtained by averaging the PR scores of all the entities appearing in the document, that is:

$$PR(D) = \frac{1}{e}\sum_{i=1}^{e} PR(d_i) \tag{2}$$

where, e is the number of entities in document D, and $d_1, d_2, ...d_e$ are the entities in the document. Since the PR method does not consider user interests, the ranking is identical for all users.

The second method, is a variant of Personalized Page Rank (PPR). The motivation behind PPR, is that while the web surfer can still jump to a random page, it is (more) likely to jump to a page according to his or her unique preferences according to a normalized vector that encodes these preferences. In this method, the recommender system recommends the items with the highest average PPR score.

The third method takes into account both PR and PPR. Both PR and PPR have drawbacks. The problem with the PR method is that it does not take the user's preferences into account at all, and assumes all users are the same. While PPR is personalized, it ignores the fact that entities that get a high PPR score, may obtain such a score not necessarily because they are highly related to the user interests, but because that they appear to be "important" in the graph in general, i.e., have a high PR score in the first place. For example, assume that the entity "Sunday" appears in a document. This entity is likely to receive a high PPR score since many entities are connected to "Sunday" and therefore, it is very likely that many of the user's interests are either directly connected to "Sunday" or connected via other entities. However, since it is so highly connected, it is also likely to receive a high PR score. The same argument holds for an entity that receives a low score, but may be quite relevant to the user. For example, if an entity is connected only to one other entity, but that other entity happens to be in the user's interest set. Therefore, to overcome these difficulties, RERA uses PR to *normalize* the PPR score and the score for RERA is given by PPR/PR. RERA recommends the items with the highest average PPR/PR score. Hence, the score RERA gives a document is obtained by:

$$RERA(D) = \frac{1}{e}\sum_{i=1}^{e} \frac{PPR(d_i)}{PR(d_i)} \tag{3}$$

For all PageRank based algorithms, due to the size of the data, rather than using the standard PageRank algorithm, we use the streaming PageRank algorithm [27]. We do this for several reasons, first, in the streaming PageRank algorithm there is no need to store the entire transition matrix in memory, but only the page rank vector. Secondly, using the streaming PageRank algorithm all updates are local and the converge rate is much faster than when using the standard PageRank algorithm.

5. Experimental Evaluation

We tested the efficiency on RERA on the MovieLens 1M Dataset [26]. Since the dataset does not contain any movie synopsis, these were crawled from IMDB. We split the movies rated by the users into train and test sets and measure the precision each of the methods reaches when recommending 10 movies. We specifically use precision at 10 since recommender systems are usually limited to a fixed number of recommendations that the user can view (without a need to scroll down the page). Therefore, a recommender system would try to maximize the number of items that the user would like within that limit.

5.1. Experimental Setup

We considered the three methods mentioned above, i.e., PR, PPR and RERA, along with two additional baselines: Bag-of-Entities (BoE), which considers only exact matches between the entities in the document and the user's interest, and the commonly used baseline of Bag-of-Words (BoW), which uses all words in the document (both for user interest and for proposed content), as opposed to using only the user entities. For BoW, we processed the documents for tokenization, stop-word removal and stemming. For extracting NELL entities from documents we used the NELL Knowledge On Demand as described in Section 4. In order to determine the hyper-parameters of both PR and PPR, we performed a grid search on testing the following settings for PR: $\alpha \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$ and for PPR: $\alpha = \{0.95, 0.9, 0.85, 0.8, 0.75\}$, $\beta \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$ and $\gamma \in \{0.05, 0.1, 0.15, 0.2, 0.25\}$. We picked the following hyper-parameters which performed best on a validation set (this set was not used in evaluation). For the PR approach, α was set to 0.1 whereas For PPR, α was 0.8, β was 0.15 and γ was set to 0.05. In the MovieLens dataset, we considered any movie that received 4 or 5 stars as a movie that the user liked and filtered users such that they liked or rated at least 50 movies. The input to the different methods (the training set that was used to compute the users' interests) varied from only 5 movies, to 30 movies. All the movies were selected randomly from the set of movies the user liked. Precision at 10 values were averaged over 10 random trials.

In addition to predicting which movies would be liked by the user, we also tried to predict which movies were *rated* by the user (movies that may have received any number of stars). The train and test set to this method was similar to the above, but sampling from all movies the users have rated.

5.2. *Experimental Results*

Figure 2 presents the precision of each of the methods when recommending 10 movies, i.e., the fraction of movies in the top 10 scores that appear in the test set as those that were liked by the user. As depicted by the figure, RERA (PPR/PR) outperformed all baselines in all conditions. PPR comes in second, and PR third, with the two baselines (BoE and BoW) way behind. Similar results are obtained when these methods were used in order to predict whether the user *rated* a movie (see Figure 3). As expected, the performance of all methods generally increases as the number of movies in the training set increases. The only exception is the PR method, which does not take into account the user history at all. Note that RERA can benefit from knowledge that the user has previously liked as

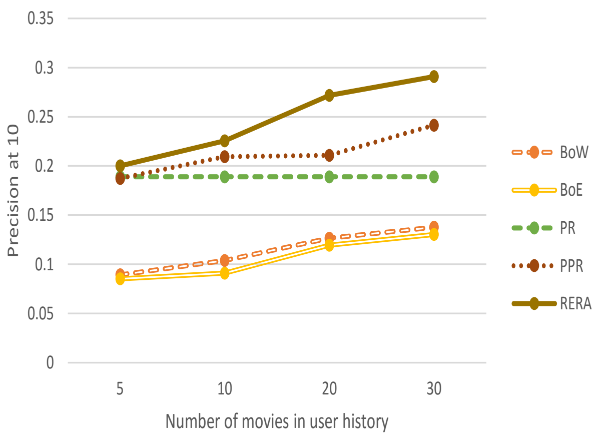

Fig. 2. Precision at 10 for each of the methods, when predicting whether the user would like a movie, with movies in user history (training set) between 5 to 30

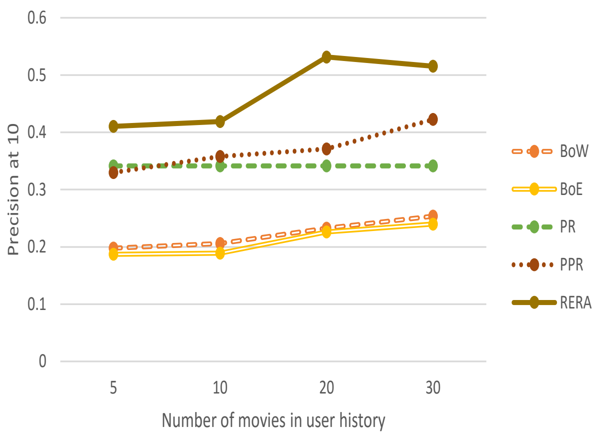

Fig. 3. Precision at 10 for each of the methods, when predicting whether the user has rated a movie

little as 5 movies. We believe that had the user explicitly provided and selected these 5 movies, RERA's precision might have been even better, as the user would have provided movies that better indicate her preferences than movies selected by random.

Another interesting result that can be observed from Figures 2 and 3 is that BoW had only a very slight advantage over BoE. This is very encouraging, as this might hint that the entities in the document do indeed capture the essence of it, and therefore, recommending content based on the entities alone is as good as using all the words in the document.

We conclude this section with an example set of recommendation given by RERA to one of the users (in the data-set). RERA received an indication that the user has liked the following movies: L.A. Confidential, Bringing Up Baby, Psycho, My Fair Lady and Butch Cassidy and the Sundance Kid. The top movies in RERA's set of recommendations include the movies, Mary Poppins and Picnic, and Ran. RERA believes that the user would like Mary Poppins as the entities "children", "family", and "English", which appear in the description of Mary Poppins, have received high RERA scores. For the movie Picnic, the entity "children", as well as the entities "college" and "envy", which appear in Picnic, all received high RERA scores. For Ran, the most important entities found by RERA included "king" "power" and "madness".

5.3. *Discussion*

Our results indicate that a recommender system can more than double its precision, by using a knowledge graph, which includes millions of entities along with the relationships between them. Our method, is a privacy-aware recommender system, and as such requires information only from the user it is currently serving. Our results may indicate that companies can benefit from recommender systems without saving sensitive information on their users. Instead of using privacy invasive methods to recommend items to one user based upon the behavior of another user, these companies may base their recommendations solely on information that the user voluntarily reveals by feeding this information into a knowledge-graph such as NELL. Such a privacy-aware recommender system may increase the trust it receives from its users, which in-turn may reveal additional information in order to obtain more relevant recommendations without feeling that their privacy is being computerized. Our method can also be applied for solving the cold-start problem. A collaborative-based recommender system, that did not collect enough information about different users to provide meaningful recommendations, can provide recommendations based upon RERA, until it gathers enough information to use a collaborative-based approach.

While we have shown that RERA more than doubles the precision in comparison to our baselines, there may seem to be a lot of room from improvement, since RERA achieves a precision rate between 30% to 50%. We argue that RERA's performance should only be compared to the performance of a baseline method, since these methods are used to predict whether a user has liked (or rated) a movie. This is a difficult task, since there is a lot of randomness considering which movies the user has chosen to rank. For example, if RERA predicts that a user would like a movie, but the user did not rank that movie at all, we consider it as a failure (even though in real-life the user may have actually liked the movie, but simply did not rate it). (This is because the data-set is too sparse to be used for evaluating only movies rated by the users.)

Surprisingly, the PageRank method, which does not depend at all on the user interests, performed relatively well. This implies that movies that contain entities that are "important" (receive high PageRank scores) in the knowledge graph, are often, in practice, ranked higher. This is very surprising, since movies are only a small fraction of the information that NELL is trained on, and therefore, we did not expect well connected entities in NELL's graph to be such a good indicator as to whether users will like different movies. Since PR is a "one-size-fits-all" approach, its performance is limited, however, it may be used for in case no information on the user's preferences or history is present at all.

Although the trends of all methods were preserved between the two experiments (Figures 2 and 3), the precision values were almost double when predicting whether the user only *rated* a movie (rather than *liked* it). While movies liked by each user are a sub-set of the movies rated by that user, which results in a higher precision (but a lower recall), this accounts only for a rise of 60% (since users are more likely to rate movies with 4 or 5 stars than 1 or 2 stars). We therefore, can conclude that it is easier to predict whether the user has rated, or in fact watched a movie. This is not surprising, since the user is likely to watch and rate movies that she is interested in (and therefore the entities that appear in them are closely related to the user's interests). This may hint that it might be beneficial for a recommender system to split the problem of recommending content that the user likes to two. In the first step it could try to predict whether the user had rated (or would watch) a movie, i.e., whether a movie lies in the user's field of interest, and then, in the second step, predict whether the user would like it.

6. Conclusion and Future Work

In this paper we introduce RERA, a recommender system that uses an enhanced NELL knowledge graph consisting of entities and relations between them to recommend content to users. RERA finds the NELL entities that are of interest to the user and the NELL entities that are mentioned in the proposed content.

RERA uses a novel enhanced version of the personalized page rank algorithm, to determine how well connected these sets of entities are in order to rank the relevance of the proposed content. We show that RERA outperforms other baseline methods.

Although our experiments were conducted in the movie domain, this is merely because this was our source of data. Our approach is general and can work in any domain. Our method could be used also for search. If a search-engine is aware of the user's general interest, RERA can help by providing an additional input for the ranking algorithm. We are currently working on deploying RERA in a news recommendation application for mobile devices (building upon Yahoo! news feed). RERA will allow the user to provide a short paragraph explicitly stating what she is interested in. RERA will then extract NELL entities from this paragraph, and using our normalized version of personalized PageRank, RERA will identify which news articles should be shown to the user once a new stream of news articles is available. If approved by the user, RERA will track articles that are read or skipped by the user. RERA will rely on the amount of time the user spends on each article, as well as indications to whether the user has clicked on any articles in order to receive further information on that news article. Furthermore, when combined with LIA [5], these user interests can also be used for playing music, games, or anything else the user may want to teach LIA.

7. Acknowledgment

This research was supported by Yahoo! as part of the InMind project [4], and by Samsung.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, June 2005.

[2] E. Aïmeur, G. Brassard, J. M. Fernandez, and F. S. M. Onana. Alambic: a privacy-preserving recommender system for electronic commerce. *International Journal of Information Security*, 7(5):307–334, 2008.

[3] A. Azaria, A. Hassidim, S. Kraus, A. Eshkol, O. Weintraub, and I. Netanely. Movie recommender system for profit maximization. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 121–128. ACM, 2013.

[4] A. Azaria and J. Hong. Recommender system with personality. In *Proceedings of the 10th ACM conference on Recommender systems*. ACM, 2016.

[5] A. Azaria, J. Krishnamurthy, and T. M. Mitchell. Instructable intelligent personal agent. In *AAAI*, pages 2681–2689, 2016.

[6] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, Dec. 2007.

[7] J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez. Recommender systems survey. *Know.-Based Syst.*, 46:109–132, July 2013.

[8] J. Callan, M. Hoy, C. Yoo, and L. Zhao. Clueweb09 data set, 2009.

[9] L. Candillier, F. Meyer, and M. Boullé. Comparing state-of-the-art collaborative filtering systems. In *Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition*, MLDM '07, pages 548–562, Berlin, Heidelberg, 2007. Springer-Verlag.

[10] S. Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. In *Proceedings of the 16th international conference on World Wide Web*, pages 571–580. ACM, 2007.

[11] M. Eirinaki, M. Vazirgiannis, and I. Varlamis. Sewep: using site semantics and a taxonomy to enhance the web personalization process. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 99–108. ACM, 2003.

[12] Z. Huang, W. Chung, T.-H. Ong, and H. Chen. A graph-based recommender system for digital library. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 65–73. ACM, 2002.

[13] K. Lee and K. Lee. Escaping your comfort zone: A graph-based recommender system for finding novel recommendations among relevant items. *Expert Systems with Applications*, 42(10):4851–4858, 2015.

[14] X. Li and H. Chen. Recommendation as link prediction in bipartite graphs. *Decis. Support Syst.*, 54(2):880–890, Jan. 2013.

[15] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014.

[16] R. Liu and Z. Jin. An improved graph-based recommender system for finding novel recommendations among relevant items. 2015.

[17] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[18] H. Ma, I. King, and M. R. Lyu. Mining web graphs for recommendations. *IEEE Trans. on Knowl. and Data Eng.*, 24(6):1051–1064, June 2012.

[19] R. V. Meteren and M. V. Someren. Using content-based filtering for recommendation.

[20] T. M. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi Mishra, M. Gardner, B. Kisiel, J. Krishnamurthy, et al. Never-ending learning. In *AAAI'15*, 2015.

[21] R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204. ACM, 2000.

[22] 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. *CBRecSys*, 15:10–13, 2015.

[23] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 85–92. ACM, 2013.

[24] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[25] A. Passant. dbrec-music recommendations using DBpedia. In *International Semantic Web Conference*, pages 209–224. Springer, 2010.

[26] J. Riedl and J. Konstan. Movielens dataset, 1998.

[27] A. D. Sarma, S. Gollapudi, and R. Panigrahy. Estimating pagerank on graph streams. *Journal of the ACM (JACM)*, 58(3):13, 2011.

[28] J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pages 158–166, New York, NY, USA, 1999. ACM.

[29] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[30] A. Segal, Z. Katzir, Y. Gal, G. Shani, and B. Shapira. Edurank: A collaborative filtering approach to personalization in e-learning. In *Proceedings of the 7th conference on Educational Data Mining*, 2014.

[31] A. Stefani and C. Strappavara. Personalizing access to web sites: The siteif project. In *Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT*, volume 98, pages 20–24, 1998.

[32] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, Jan. 2009.

[33] G. Tian and L. Jing. Recommending scientific articles using bi-relational graph-based iterative rwr. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 399–402. ACM, 2013.

[34] Y. Zhang, R. Jin, and Z.-H. Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.