Comparing direct and indirect approaches to weighted SimRank for Amazon Movie recommendations

Sevket Melih Zenciroglu
smzen@kth.se

Federico Jimenez Villalonga
fjv@kth.se

Koray M. Kaya
kkaya@kth.se

Philip Wester
phwe@kth.se

1 Abstract

In this work, we explore the topic of link prediction for bipartite graphs. The dataset is a list of movie reviews coming from the Amazon website between 1997 and 2012, from which we extracted the bipartite graphs used in the experiments. In the graph, a node can either be a movie or a user, the edges are represented by the reviews and the weights are given by the score of the review. Two experiments were run, both used the SimRank algorithm. The first one predicted the creation of a review based on the SimRank distance between the user and the movie. In the second experiment, the SimRank algorithm was used to find the group of the users most similar to the target-user (i.e. the user to which we want to recommend a movie to). The movies were then ranked by the percentage of similar users who already reviewed them. The top ones were the results of this recommendation system. Even though the average degrees of the graphs were very low, we believe that the overall results of our recommendation system is in pair with similar models.

2 Introduction

With the fast growth of internet services and internet shopping, the need for good recommendation systems increases. Strong algorithms, such as Amazon's or Netflix's recommendation engines, has helped customers find products that suit them and the company to profit. The goal of this project is to, with the help of a data set, recommend movies to users.

The data set comes from Amazon and is based on earlier movie reviews. Each entry in the data set contains a set of attributes. The attributes used by this project are reviewer ID, movie ID, rating, and date. The algorithm needs the recipient of the recommendation to be logged in the data set in question. If this requirement is fulfilled to a sufficient extent, the algorithm will be able to recommend a number of new movies for the recipient to watch and enjoy.

2.1 Bipartite graphs

One of the challenges with movie recommendations is that the graph is bipartite. This guarantees that there are no direct links between reviewers/users and the absence of triangles in the graph. Because of this, the clustering coefficient is constantly 0. Furthermore, the structure of the graph demand a separation of similarity between types, e.g. a user should in the end be recommended movies and never be recommended another user. While unipartite graphs might be considered to be a more thoroughly explored topic than bipartite (1; 2; 3; 4; 5), there still exist a considerable number of works with bipartite graphs (6; 7; 8; 9).

This was solved by using SimRank, which is used to calculate the similarity score between users and users and movies and movies to ultimately recommend movies to users. The SimRank intuition is also independent of the clustering coefficient and the creation of triangles in the graph.

3 Related Work

This project creates movie recommendations based on a previous data set. This is very close to the common link prediction problem and similarity measuring. It is an area that has been quite thoroughly explored, even in bipartite graphs, where some of the related works are SimRank and Supervised random walks on social graphs.

3.1 SimRank

One of the most famous and similar work is the SimRank algorithm (10), which is an extension of the PageRank algorithm. The base intuition of SimRank is that "similar objects are related to similar objects". With this assumption, SimRank uses the structure of k-partite graphs to draw conclusions about similarities between objects of the same disjoint set, even though there can never be any direct links between. For example if person A buys product a, b, and c. A should be more similar to person B who buys products a, d, and c than person C who buys products e, f, and g. This can also be extended with similarities between products, product a and b should be more similar than a and f since a and b are more often bought together.

SimRank will rank each pair of nodes a, b with a SimRank score $s(a,b) \in [0,1]$, where only the node itself can achieve the SimRank value 1 with itself, $a \neq b \leftrightarrow s(a,b) < 1$. The calculation of $s(a,b)$ is based on the similarity between the links of a and b:

$$s(a,b) = \begin{cases} s\colon \{1,\ldots,n\} \to \{0,\ldots,1\} \\ s(a,b) = \frac{C}{E(a)E(b)} \sum_{i=1}^{E(a)} \sum_{j=1}^{E(b)} s(E_i(a), E_j(b), \text{for a } \neq \text{ b} \\ s(a,b) = 1, \text{for a = b} \end{cases}$$

Where $E(a)$ denotes the degree of a, $E_i(a)$ denotes the i:th neighbor of a and C is a constant.

There are multiple methods that can be used to calculate the SimRank over an entire graph, and in our case, it would be by using a weighted random walk where the weights represent the normalized review scores.

3.2 Supervised Random Walks on Social graphs

In (11), Backström et al. create a function so that they, in a supervising way, can assign edge weights that promote a random walk in their interest. This function does not solely use the graph structure itself as a parameter but also use features that have been labeled to the edges and nodes. As an example, the function might take the node feature "age" as a parameter when assigning an edge weight between two nodes.

This paper will not use a complex supervised method to assign edge weight. Instead, it will be a very simple method that assigns edge weights directly proportional to the corresponding review.

3.3 Link prediction on bipartite graphs

In (12), Zhong-you Pei, Chun-heng Chiang and Wen-bin Lin create personalized recommendations from a bipartite graph with the help of Random Walks. Zhong-you Pep et al. use a PageRank method very similar to SimRank, to extract the top K recommendations for each user.

While it is unclear exactly how this method differs from SimRank, since the paper does not mention the relation to SimRank, the method is described in more practical terms and can, therefore, be said to be more closely related to this project than the original SimRank paper.

4 Method

Here we present two different models for recommending one or more movies to a user. Both proposed models have some features in common:

- Both models pick two sets of data, one for modeling and one for testing/validation, which is there for all the experiments.
- Both models make use of the random walk algorithm having review-scores as weights and teleportation to the starting node with a probability ß.
- Finally, the sorting of the nodes is made regarding their ranks from the random walk.

They both look at past behaviors to predict future edges. One method examines the connection between a user and its movies while the other exploits other user-movies interaction first to recommend the most similar movies.

4.1 Dataset

The original dataset consists of almost eight million movie reviews from the Amazon website made between 1997 and 2012. A total of 890,000 different users and 250,000 movies appear in the dataset. It can be downloaded as a .txt file from here [1].

Each element or review contains 8 variables, namely: the movie ID, the reviewer ID, the name of the reviewer, helpfulness of the review (judged by other users), a score (from 1 to 5), the time at which the review was posted, a summary of the review and a text comment from the reviewer. In order to avoid carrying superfluous information to later stages all variables except movie ID, reviewer ID, time and score were removed. These four attributes are indeed enough to identify a single review: a user can only post one review per movie and each review carries only one score.

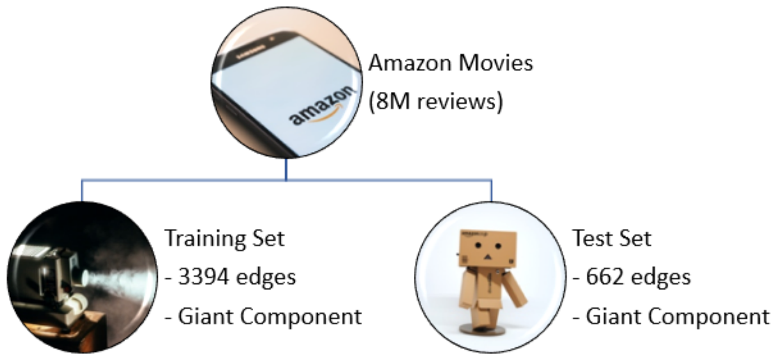

Figure 1: How the data was extracted.

4.1.1 Creating the Graphs

A part of the main dataset was ordered by the date at which the reviews were posted from 2009. Afterward, two subsets were extracted. The first one comprising of 5000 reviews became the model-set having data from January 2009 including some data of September 2009. The second one of 1200 reviews became the test-set which covers the rest of the year 2009. Ordering by time, however, led to an imbalance between the number of reviews per movie: many movies did not have enough reviews to be clearly defined. Sometimes a movie would have only one or two reviews. This phenomenon created very sparse graphs whose biggest connected component counted less than 50% of the nodes of the graphs. Such problem is hard to solve without taking into account the totality of the dataset, and even then, it is not certain that the resulting graph would be dense enough to create bigger connected components.

[1]https://snap.stanford.edu/data/web-Movies.html

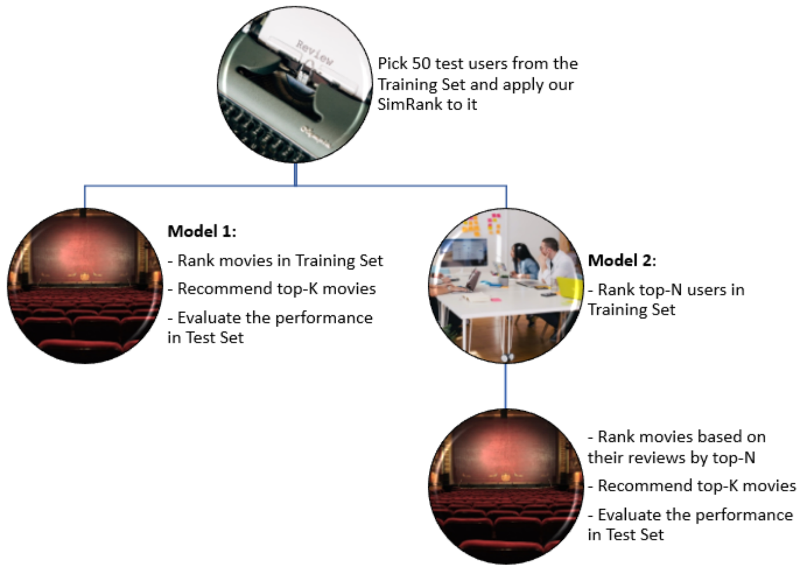
Pick 50 test users from the Training Set and apply our SimRank to it
Model 1:
- Rank movies in Training Set
- Recommend top-K movies
- Evaluate the performance in Test Set
Model 2:
- Rank top-N users in Training Set
- Rank movies based on their reviews by top-N
- Recommend top-K movies
- Evaluate the performance in Test Set

Figure 2: The two algorithms.

4.2 Model 1

In the first evaluation method, we implemented the SimRank algorithm to find the best recommendation. For instance, we executed the random walk with weights and a certain number of steps on a user. At the end of the walk, we sorted the movies regarding their scores in the rank vector. This was repeated for D different users. Finally, the Top-K movies were recommended in dataset-1 for the D different target-users, as their next movies they will review. Ultimately, these recommendations were checked with the true values in dataset-2 and the precision was calculated by averaging the successful results for those D users. Moreover, the AUC was calculated to evaluate the overall performance of the model in all movies, not only considering the top-K movies recommended. K was set to 1 and D was set to 50.

4.3 Model 2

In the second evaluation method, the SimRank algorithm was used to find the most similar users (top-N similar user, i.e. N=20) for each of our target-users. The top-N user array was used in dataset-1 to calculate for the movie-review ratios. For example, assume we have 50 movies in dataset-1, movie-1 was reviewed by 30% of those top-N users, movie-2 was reviewed by 40% of top-N users, and movie-3 was reviewed by 5% of top-N users. We used the top-K recommendation for movies by having K=1. So, in this case, movie-2 was recommended as the user will be very likely to review this movie in dataset-2 (the last/second part of the year 2009). This procedure was repeated for D different users. In all cases, movies have already been reviewed by the user in the recommendation target were removed from the movie list, so that we will not recommend a movie which has already been reviewed by the user.

5 Results

Our research compared two methods of suggesting movies through SimRank. Since the methods do not perform well in the same parameter ranges, we will be using the best 5 results of each method on the same dataset and compare them. We chose the AUC and precision metrics for performance evaluation since they are widely used metrics for link prediction.

5.1 SimRank to pick movies

We used SimRank to measure distance between the user and movies, then suggested the closest movie.

Table 1: The 5 best results for recommending one movie through SimRank distance

Walk steps	Top neighbor	n Test Users	AUC	Precision
30	20	65	0.7186	0.123
40	50	50	0.7147	0.12
30	20	65	0.7137	0.11
30	20	50	0.7008	0.12
30	50	50	0.6965	0.12

5.2 SimRank to pick reference users

We used SimRank to measure the distance between the user and other users and suggested the most commonly liked movie among the closest users.

Table 2: The 5 best results for recommending one movie based on users with closest SimRank distance.

Walk steps	Top neighbor	n Test Users	AUC	Precision
40	50	50	0.7437	0.2
40	20	50	0.7166	0.22
40	50	65	0.7142	0.18
30	50	50	0.7142	0.12
20	50	50	0.7064	0.18

6 Conclusion

Looking at the two approaches to SimRank based link prediction, we see that one gives slightly better results. Our two approaches were either directly suggesting a movie to a user with SimRank, or selecting similar users to our user with SimRank and suggesting a movie based on their collective preferences. The second approach gives consistently better results and precision than the first approach. This might be due to the fact that the second approach is a more social approach and can be compared to filtering out dissimilar users from the graph and after that running a SimRank as in the first method, effectively pre-processing the data through SimRank.

We see good results from our experiments in an expected range. Our method is quite simple compared to the related work, yet also robust. We see that our results fall into an average range for the AUC value while our precision is lower than average (11; 12). We assume that the low precision is due to the small average degree, which will be discussed below. The SimRank method constitutes a foundation for most of the related work which explains the average results. Hence, even in its simplest form, SimRank makes for an acceptable suggestion method. It would although be very useful to compare our method to a random recommendation method, giving us a baseline to our results. Furthermore, in the subject of link prediction we have more leniency in performance, since if we recommend few good movies in a list of 10 the user will likely be able to overlook the poor suggestions.

The main difficulty in interpreting the results stems from the low degree of the user nodes, ie. the low amount of reviews per user. This naturally leads to a high TPR since it is likely that we pick the few movies the user has reviewed in our large list of suggestions. E.g. if the user has reviewed 2 movies and one happens to be in the test set, and we suggest 10 movies, we only need to correctly suggest 1 movie for the TPR to be 100% and the precision 1/10. Something that would have been unlikely to occur in a more dense graph.

It is also hard to see if our suggestions were correct. Although all the suggested movies can be good, since the average degree is so low we will still suffer from a low precision capped at $\frac{1.3}{\text{recommended movies}}$ in the case that more than 1 movies is suggested. To put it in more generic terms, the precision value is limited to $\frac{\text{average degree}}{\text{recommended movies}}$ if the average degree is bigger than the number of the recommended movies. In other words, for a graph having the average degree of 4, there is no way to achieve precision 1 if we go for 10 recommendations. We can only achieve precision 1 with 4 or a lower number of recommendations.

6.1 Future work

When implementing what variable to use as the weight of the edges, we used only the scores of the reviews. However, the users that are more active and review more movies than others were graded like the inactive users who do not post as many reviews. As a further development, the activity rate of a user can be considered while constructing the weights. A similar case is valid for the movies. Some movies are more popular than others, either in a good or a bad way. Some receive a significant number of reviews in a short period of time, some others have a more linear distribution. This popularity indicator can also be included while evaluating the weights and see if it could result in a notable improvement in the performance.

It would also be interesting to filter the data before running the algorithm, or even use another data set. We see that the main concern in our tests is that the average degree is very low, making it hard to evaluate. The users with a single review also do not contribute much, they simply detour the random walker. To prevent this, we could in the future only use data-points with a sufficient degree so that we can run more satisfying tests. This problem can alternatively be solved by using another data set with a higher average degree.

References

[1] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM (JACM)*, vol. 46, no. 5, pp. 604–632, 1999.

[2] T. H. Haveliwala, "Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search," *IEEE transactions on knowledge and data engineering*, vol. 15, no. 4, pp. 784–796, 2003.

[3] R. Lempel and S. Moran, "The stochastic approach for link-structure analysis (salsa) and the tkc effect," *Computer Networks*, vol. 33, no. 1-6, pp. 387–401, 2000.

[4] Y. Liu, B. Gao, T.-Y. Liu, Y. Zhang, Z. Ma, S. He, and H. Li, "Browserank: letting web users vote for page importance," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 2008, pp. 451–458.

[5] B. Gao, T.-Y. Liu, W. Wei, T. Wang, and H. Li, "Semi-supervised ranking on very large graphs with rich metadata," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 96–104.

[6] X. He, M. Gao, M. Kan, and D. Wang, "Birank: Towards ranking on bipartite graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 57–71, 2017.

[7] H. Deng, M. R. Lyu, and I. King, "A generalized co-hits algorithm and its application to bipartite graphs," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 239–248.

[8] X. Li, M. Zhang, Y. Liu, S. Ma, Y. Jin, and L. Ru, "Search engine click spam detection based on bipartite graph propagation," in *Proceedings of the 7th ACM international conference on Web search and data mining*, 2014, pp. 93–102.

[9] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos, "Relevance search and anomaly detection in bipartite graphs," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 2, pp. 48–55, 2005.

[10] G. Jeh and J. Widom, “Simrank: A measure of structural-context similarity,” in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’02. New York, NY, USA: Association for Computing Machinery, 2002, p. 538–543. [Online]. Available: https://doi-org.focus.lib.kth.se/10.1145/775047.775126

[11] L. Backstrom and J. Leskovec, “Supervised random walks: predicting and recommending links in social networks,” in *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011, pp. 635–644.

[12] Z. Pei, C. Jiang, and W. Lin, “Random walks on the bipartite-graph for personalized recommendation,” 01 2013.

A Contributions

The whole group contributed in reading papers, algorithm development and analyzing the results. The data collection was also a group effort. Melih did most of the code implementation of the algorithm, tests and optimizing, while the others helped. The rest of the group made the presentations and wrote the report which Melih also contributed to.