Procedia Computer Science

Volume 80, 2016, Pages 2205–2210

ICCS 2016. The International Conference on Computational Science

Learning Vector-Space Representations of Items for Recommendations using Word Embedding Models

Balaji Krishnamurthy[1], Nikaash Puri[1], Raghavender Goel[2*]
[1]*Adobe Systems, Noida, India.*
[2]*Indian Institute of Technology, Delhi, India.*

kbalaji@adobe.com, nikpuri@adobe.com,raghavendergoel@gmail.com

Abstract
We present a method of generating item recommendations by learning item feature vector embeddings. Our work is analogous to approaches like Word2Vec or Glove used to generate a good vector representation of words in a natural language corpus. We treat the items that a user interacted with as analogous to words and the string of items interacted with in a session as sentences. Our embedding generates semantically related clusters and the item vectors generated can be used to compute item similarity which can be used to drive product recommendations. Our method also allows us to use the feature vectors in other machine learning systems. We validate our method on the MovieLens dataset.

Keywords: Recommendation systems, Collaborative Filtering, Item vectors, Word Embeddings

1 Introduction

The goal of a Recommendation System is to generate meaningful recommendations to users for items or products that might interest them. Some common examples include suggestions on e-commerce sites, or movies on Netflix. Collaborative Filtering (CF) is a subset of Recommendation algorithms that seeks to exploit users' interaction as well as their explicit product ratings in order to predict the rating for a user on an unseen product or to predict the propensity of a user to consume an item, for example - buy a product or view a video-content.

Memory based Collaborative Filtering or neighborhood search exploits item-item similarity or user-user similarity. The intuition here is that we determine users that are similar, and then proceed to recommend items to a user based on what similar users have liked. Further, CF algorithms are also

[*] Work done during internship with Adobe Systems

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2016

doi:10.1016/j.procs.2016.05.380

useful in determining item-item similarity measures. Intuitively, given an item, we wish to find items that are similar to the given item, based on user interactions.

Other set of CF algorithms are based on latent factors models. These can be generative probabilistic models like LDA, PLSI etc. which can be used, for example to find hidden topics that explain occurrences of words in documents. An interesting variation of latent factor model is matrix factorization where a sparse user-item matrix is factorized to find user-latent factors and item-latent factors. Since the predictions made using matrix factorization are accurate and useful practically, so it is important to also include matrix factorization in the set of recommendation algorithms (where final recommendations are a combination of output of several algorithms)

We present a method of generating item recommendations by learning item feature vector embeddings. Our work is analogous to approaches like Word2Vec or Glove used to generate a good vector representation of words in a natural language corpus. We treat the items that a user interacted with as analogous to words and the string of items interacted with in a session as sentences. Our embedding generates semantically related clusters and the item vectors generated can be used to compute item similarity which can be used to drive product recommendations.

The Paper is organized as follows. Section 2 discusses related work in this area. Section 3 describes the approach followed. In section 4 we cover our experiments and the results we've obtained. Finally in section 6 we conclude with a discussion of future work.

2 Related Work

The problem of embedding items in a low dimensional space has been largely overlooked. Most approaches focus on learning low dimensional embeddings of users and items. Matrix Factorization techniques [1, 2] factorize the user-item space and generate vector representations for each of these users and items. These representations form the basis of determining user-item affinities through a series of vector operations. [4] Describe a global log-bilinear regression model that combines the advantages of matrix factorization and local context window methods to generate word embeddings in a low dimensional space. There have been significant strides in neural embedding methods that have led to improvements in the state of the art Natural Language Processing (NLP) capabilities [3, 6, 7]. Item-Item similarities form the cornerstone of several recommender systems. [5] Look into different techniques for computing item-item similarities and different techniques for obtaining recommendations from them.

3 Our Approach

We generate item recommendations by learning item feature vector embeddings. Our work is analogous to approaches like Word2Vec or Glove used to generate a good vector representation of words in a natural language corpus. We treat the items that a user interacted with as analogous to words and the string of items interacted with in a session as sentences. This 'corpus' of items is then used as an input to a word embedding algorithm. The algorithms then learn item vector representations that capture the relationship between items. The similarity between items is captured in the representation of the item vectors and is used to build an item similarity matrix. We then use these item-item similarities to generate recommendations for users based on their browsing history. These feature vectors could alternatively also be employed as feature vectors in other machine learning

systems. We have experimented with both GloVe and Word2Vec to learn item embeddings with good results.

3.1 Learning Item Vectors

GloVe (Global Vectors for word representation) [4] describe a global log-bilinear regression model that makes use of the word-word co-occurrence matrix along with local context window methods to generate word embeddings in a low dimensional space. The GloVe model has several advantages that make it suitable for the task of learning item representations. Chief among them being that it efficiently leverages statistical information by training only on the nonzero elements in the item-item co-occurrence matrix, rather than on the entire sparse matrix or on individual context windows in a large corpus. Further, the representation deduction process in GloVe treats the distance between words to determine their relative similarity. Intuitively, this makes sense since items that are viewed consecutively in a session are likely to be more similar than items that are separated by a larger number of items within a session. As a simple illustration, consider two session streams. The first being 'I1 I2 I3 I4' and the other being 'I5 I6 I4 I1 I7'. Now, simply observing the two clickstreams we can intuitively we can deduce that I2 and I3 are more similar than I5 and I7. This kind of relationship is captured by the algorithm in GloVe.

Word2Vec [3] consists of two distinct models (CBOW and skip-gram), each of which defines two training methods (with/without negative sampling) and other variations, such as hierarchical softmax. Both of CBOW and skip-gram are shallow 2 layer neural network models. Experiments in [3] show that the skip-gram model with negative sampling (SGNS) outperforms the other variants on analogy tasks and is hence the recommended algorithm for text corpus. However, in our approach we use the CBOW model since it more intuitively captures our problem domain.

In a typical CBOW implementation the neural network is trained to predict the central word given the words that occur in a context window around it. The word representations learnt in such a way is sensitive to the sequence of the words (or items) in the context. However, in our problem domain it makes sense to learn these embeddings in an order agnostic manner. So, to make the model less sensitive to these orderings we include a number of random permutations of the items in a user session to the training corpus.

4 Results

For our experiments we have used the MovieLens 10M Dataset which contains10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. Tables 1 and 2 detail the results obtained using GloVe. For learning the GloVe vectors we have used a window size of 15 and a dimensionality of 50. The similarity between items is computed as a dot product of the movie vectors. At the time of prediction, we weigh the movie similarity scores with the ratings the user has given these movies in order to rank the set of predicted movies.

The metric we have used to test our approach is the recall@20 measure. The measure is computed as follows. For each user, we remove the last movie that the user has seen. We then use the set (of size K) of previously seen movies to predict 20 movies that the algorithm believes the user is most likely to watch. If the removed movie is present in this list of 20 recommended movies we count the prediction as a success, else a failure. The metric is then simply the total percentage of successes over all users. The recall@20 measure is a more realistic metric for comparing recommendation algorithms as it

captures user behavior in a more explicit manner than measures such as the RMSE in rating prediction.

For the purpose of comparison we use the co-occurrence based recommendation algorithm. This approach involves building an item-item matrix where each entry in the matrix reflects the number of times two items co-occur in the user session data. At prediction time, for each item, we predict the top 20 items that co-occur with that item as recommendations. Co-occurrence is a powerful baseline and it captures several aspects of item similarity. In fact, it outperforms matrix factorization techniques such as ALS when using the recall@20 measure.

As we can see from Table 1, our approach performs significantly better than the co-occurrence approach in predicting movie recommendations. As we increase the number K (the number of previous movies being used to make recommendations) the success rate decreases. Intuitively, this suggests that the most recent movie that the user has seen is the most informative feature in

Algorithm	K=1	K=2	K=3	K=4
GloVe embeddings	54	46	42	38
Co-Occurrence	28	24	23	24
Word2Vec without permutations	24	21	19	15
Word2Vec with permutations	30	28	28	27

Table 1: Results for recall@20 using the last K videos for a time window of N<61 days

determining the movie he is going to see next. Similarly, in table 2 we show the performance of our algorithm as we vary N (difference in time between the last movie and the watched movies). Intuitively, movies that the user has watched more than a year ago are unlikely to influence his current intent. As we can see from the table, for large values of N, the accuracy diminishes slightly. This backs up the intuition that for those cases where the time difference between the removed movie and the previous K movies is very large, the previous K movies in such cases are not as good indicators of the next movie the user is going to watch.

Algorithm	N<8	N<15	N<31	N<61	N<366
GloVe embeddings	56	55	55	54	50
Co-Occurrence	26	27	27	28	22
Word2Vec without permutations	25	25	24	24	22
Word2Vec with permutations	31	30	29	30	27

Table 2: Results for recall@20 using the last K=1 videos within a time window of N days

We have also conducted several experiments using Word2Vec to learn item embeddings and generating recommendations. We use CBOW with a window size of 15 and Negative Sampling. The dimensionality of the item vectors was set to 50. The experiments were carried out on the MovieLens 10M dataset and the approach gives promising results. The results are shown in tables 1 and 2. We have trained the Word2Vec model both by permuting the input sentences and by using the input

sentences as is. In the approach, we consider 20 permutations of each sentence along with the sentence itself in training the Word2Vec model. As we can see from the tables, using permutations with the Word2Vec model improves performance and gives better results than Co-occurrence. GloVe, however significantly outperforms Word2Vec in the task of movie recommendations. Figure 1 and 2 depict the same results as Table 1 and 2 in a graphical format. It is clear from these that GloVe outperforms both Co-Occurrence as well as Word2Vec in movie recommendations.

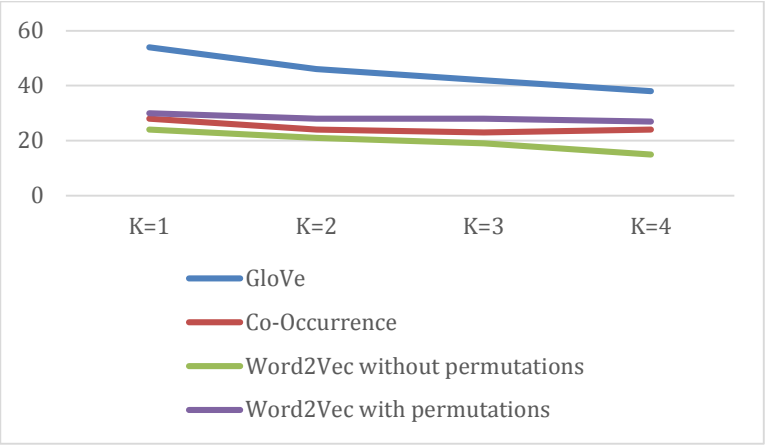

Figure 1: Algorithm performance for different values of K (past number of items used for recommendations)

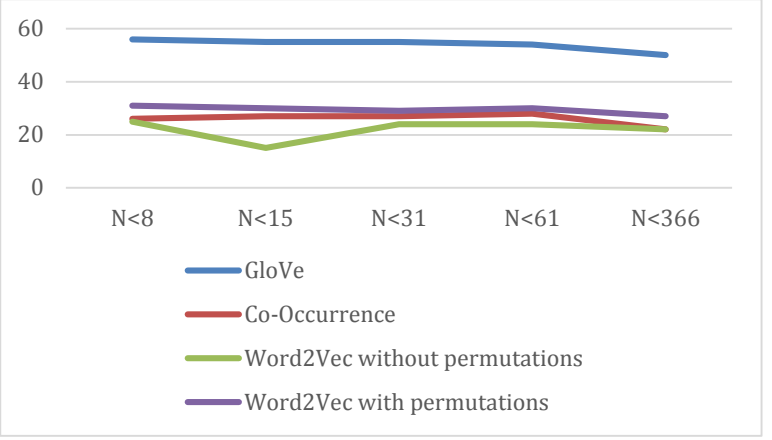

Figure 1: Algorithm performance for different values of N (time window, in days)

5 Conclusion

Our inter-disciplinary approach of utilizing methods that are highly successful in the field of Natural Language Processing (NLP) in order to generate item recommendations shows very good results. We have shown that it significantly outperforms a co-occurrence based recommendations which is a strong baseline for our measure. In the future, we plan to further the effectiveness of utilizing the user ratings in learning the Item representations.

References

[1] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer* 8 (2009): 30-37.

[2] Zhou, Yunhong, et al. "Large-scale parallel collaborative filtering for the netflix prize." *Algorithmic Aspects in Information and Management*. Springer Berlin Heidelberg, 2008. 337-348.

[3] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

[4] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation." *EMNLP*. Vol. 14. 2014.

[5] Sarwar, Badrul, et al. "Item-based collaborative filtering recommendation algorithms." *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001.

[6] Collobert, Ronan, and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning."*Proceedings of the 25th international conference on Machine learning*. ACM, 2008.

[7] Mnih, Andriy, and Geoffrey E. Hinton. "A scalable hierarchical distributed language model." *Advances in neural information processing systems*. 2009.

[8] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872