Tag-Aware Personalized Recommendation Using a Deep-Semantic Similarity Model with Negative Sampling

Zhenghua Xu[*] Cheng Chen[†] Thomas Lukasiewicz[*] Yishu Miao[*] Xiangwu Meng[†]

[*]Department of Computer Science, University of Oxford, United Kingdom
{zhenghua.xu, thomas.lukasiewicz, yishu.miao}@cs.ox.ac.uk
[†]School of Computer Science, Beijing University of Posts and Telecommunications, China
{ccbupt,mengxw}@bupt.edu.cn

ABSTRACT

With the rapid growth of social tagging systems, many efforts have been put on tag-aware personalized recommendation. However, due to uncontrolled vocabularies, social tags are usually redundant, sparse, and ambiguous. In this paper, we propose a deep neural network approach to solve this problem by mapping both the tag-based user and item profiles to an abstract deep feature space, where the deep-semantic similarities between users and their target items (resp., irrelevant items) are maximized (resp., minimized). Due to huge numbers of online items, the training of this model is usually computationally expensive in the real-world context. Therefore, we introduce negative sampling, which significantly increases the model's training efficiency (109.6 times quicker) and ensures the scalability in practice. Experimental results show that our model can significantly outperform the state-of-the-art baselines in tag-aware personalized recommendation: e.g., its mean reciprocal rank is between 5.7 and 16.5 times better than the baselines.

Keywords

Tag-Aware Personalized Recommendation; Deep Neural Network; Deep-Semantic Similarity; Negative Sampling

1. INTRODUCTION

In the era of Web 2.0, social tagging systems are introduced by many websites, where users can freely annotate online items using arbitrary tags (commonly known as *folksonomy* [5]). Since social tags are good summaries of the relevant items and the users' preferences, and also contain little sensitive information about their creators, they are valuable information for privacy-enhanced personalized recommendation. Consequently, many efforts have been put on tag-aware personalized recommendation using *content-based filtering* [2, 8] or *collaborative filtering* [1, 9]. However, as users can freely choose their own vocabulary, social tags may contain many uncontrolled vocabularies, such as homonyms, synonyms, words in arbitrary languages, or even user-created words. This results in very sparse, redundant, and ambiguous tag information, which greatly degrades the performance of the tag-aware recommendation systems.

A solution to this problem is to apply clustering in the tag space [8]; however, clustering requests to compute the similarity between tags which is usually very time-consuming (or even intractable). Another solution is to use autoencoders; in [10], abstract representations for tag-based user profiles are first modeled by autoencoders and then used as inputs of user-based collaborative filtering to generate recommendation. Although this method is reported to achieve better performance than the clustering-based collaborative filtering method [10], it still suffers from two drawbacks: (i) the model's learning signal comes from the reconstruction error, which is not the objective of personalized recommendation, i.e., distinguishing the user's target items from the irrelevant ones; (ii) the recommendation is solely based on user profiles, so the relevance between users and items is indirectly inferred from other similar users. So its performance in personalized recommendation is damaged to a great extent.

In this paper, motivated by the above observations, we propose to address the uncontrolled vocabulary problem by using deep neural networks to map both the tag-based user and item profiles to an abstract deep feature space, where the similarities between users and their target items (resp., irrelevant items) are maximized (resp., minimized). We call the similarities on the deep feature space *deep-semantic similarities* and this model *deep-semantic similarity based personalized recommendation* (DSPR) model. DSPR has the following advantages: (i) The training objective is directly correlated with differentiating the user's target items from the irrelevant ones, so the resulting abstract features for user and item profiles are very effective representations for personalized recommendation. (ii) The relevance between users and items is directly computed using the similarities between the abstract user and item profiles.

However, to train DSPR, the deep-semantic similarities between the user in each training sample and all the candidate items have to be computed in each training run. As the number of candidate items for a recommendation system is usually very large (millions), and training deep neural networks often requires numerous training samples and many

CIKM'16, October 24–28, 2016, Indianapolis, IN, USA.

DOI: http://dx.doi.org/10.1145/2983323.2983874

training runs, training the DSPR model is computationally expensive in practice, which results in scalability problem. To address this problem, we propose to use *negative sampling* [7], to significantly enhance DSPR's training efficiency, while maintaining almost the same training effectiveness.

The contributions of this paper are threefold: (i) We propose a tag-aware personalized recommendation system that uses deep-semantic similarity-based neural networks to extract abstract and recommendation-oriented representations for tag-based user and item profiles so as to achieve a superior personalized recommendation. (ii) For scalability in practice, we use negative sampling to increase the model's training efficiency (109.6 times quicker). (iii) We provide experimental results, which show that our model can significantly outperform the state-of-the-art baselines in tag-aware personalized recommendation: e.g., its mean reciprocal rank is between 5.7 and 16.5 times better than the baselines.

2. RELATED WORK

Many systems have been proposed for tag-aware personalized recommendation on the Social Web. Content-based systems [2, 8] aim at recommending items that are similar to those that a user liked previously. Collaborative systems recommend to users items liked by similar users using machine learning techniques, such as nearest neighbor modeling [9] and matrix factorization [1]. Due to uncontrolled vocabularies, social tags are usually redundant, sparse, and ambiguous. A solution is to apply clustering in the tag space to aggregate redundant tags and reduce ambiguities [8]. But tag clustering is usually time-consuming in practise, so another solution is to use autoencoders [10].

Deep learning has been successfully applied in many search and recommendation applications, such as item recommendation [4] and Web search [6]. Similarly to our work, [4] and [6] use deep-semantic similarity models with a ranking-oriented training objective. However, these models are very different from DSPR: (i) they are not tag-aware systems and not designed to solve the redundancy, sparsity, and ambiguity problems in tag space; (ii) instead of using negative sampling, these two works intentionally assume the number of candidate items to be very small (5 in [6] and 10 in [4]) to make the model trainable. Obviously, this assumption is unreasonable in real-world situations. Negative sampling was first introduced in the NLP community to learn word representation more efficiently [7]. To our knowledge, our work is the first that applies negative sampling to enhance the training efficiency of deep-semantic similarity-based models.

3. PRELIMINARIES

A *folksonomy* is a tuple $\mathcal{F} = (U, T, I, A)$, where U, T, and I are sets of *users*, *tags*, and *items*, respectively, and $A \subseteq U \times T \times I$ is a set of assignments (u, t, i) of a tag t to an item i by a user u [5].

A *user profile* is a feature vector $x_u = (g_1^u, \ldots, g_M^u)$, where $M = |T|$ is the tag vocabulary's size, and $g_j^u = |\{(u, t_j, i) \in A \mid i \in I\}|$ is the number of times that user u annotates items with tag t_j. Similarly, an *item profile* is a vector $x_i = (g_1^i, \ldots, g_M^i)$, where $g_j^i = |\{(u, t_j, i) \in A \mid u \in U\}|$ is the number of times that item i is annotated with tag t_j [2].

Personalized recommendation is then defined as follows. For a user u, the system produces a ranked recommendation list $\tau = [i_1 \geq i_2 \geq \cdots \geq i_n]$ for all items s.t. $i_a \geq i_b$ iff $Recom(u, i_a) \geq Recom(u, i_b)$, where $Recom(u, i)$ is a recommendation function measuring how relevant item i is to u.

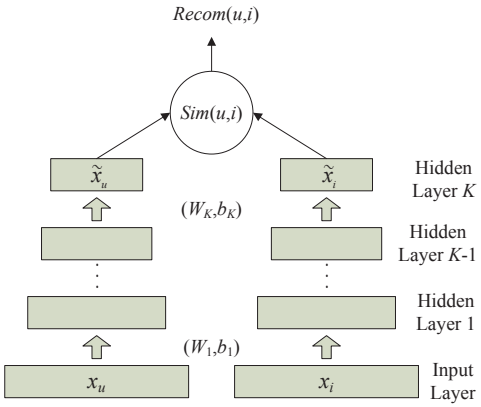

Figure 1: Overview of DSPR

4. DSPR WITH NEGATIVE SAMPLING

Figure 1 shows an overview of our *deep-semantic similarity-based personalized recommendation* (*DSPR*) model. Generally, DSPR takes the tag-based user and item profiles x_u and x_i (as defined in Section 3) as inputs of two deep neural networks with shared parameters. These inputs are then passed through multiple hidden layers and projected into an abstract deep feature space on the final hidden layer, where the similarities between the abstract representations of user and item profiles are computed. Finally, a recommendation function is used to rank items by applying the softmax function on the resulting similarities.

The motivation of sharing parameters is as follows: (i) Since both user and item profiles are modeled from the same folksonomy, they share the same tag space and same value range in each dimension; so it is reasonable to use shared parameters to reduce computational and memory costs in model training. (ii) Shared parameters make two neural networks use the same abstract feature space to describe users and items, which is beneficial for measuring their relevancy.

Formally, given the user profile x_u, the item profile x_i, a weight matrix W_1, and a bias vector b_1, the intermediate output h_1 of the first hidden layer is defined as follows:

$$h_1(u) = \tanh(W_1 x_u + b_1), \tag{1}$$

$$h_1(i) = \tanh(W_1 x_i + b_1), \tag{2}$$

where tanh is used as the activation function. Similarly, the intermediate output of the jth hidden layer h_j, $j \in \{2, \ldots, K\}$, is defined as:

$$h_j(u) = \tanh(W_j h_{j-1}(u) + b_j), \tag{3}$$

$$h_j(i) = \tanh(W_j h_{j-1}(i) + b_j), \tag{4}$$

where W_j and b_j are the weight matrix and the bias vector for the jth hidden layer, and K is the total number of hidden layers. The outputs of Kth hidden layer are the abstract feature representations of user and item profiles, denoted $\widetilde{x}_u$ and $\widetilde{x}_i$, respectively. Formally,

$$\widetilde{x}_u = h_K(u), \qquad \widetilde{x}_i = h_K(i). \tag{5}$$

Then, the similarity between a user u and an item i is measured using the cosine similarity between the abstract representations of their profiles, formally defined as

$$Sim(u, i) = \frac{\widetilde{x}_u \cdot \widetilde{x}_i}{\|\widetilde{x}_u\| \|\widetilde{x}_i\|}, \tag{6}$$

and called *deep-semantic similarity*.

Finally, to generate personalized recommendations, a recommendation function is used to measure the relevance of an item i to a user u by applying the softmax function on the resulting deep-semantic similarities; formally,

$$Recom(u,i) = \frac{e^{Sim(u,i)}}{\sum_{i' \in I} e^{Sim(u,i')}}. \quad (7)$$

As we assume that the target items of a given user are those annotated by this user, to achieve good personalized recommendations, these items should have higher recommendation scores than others. We thus conduct the model training with an objective to maximize the recommendation scores of target items; equivalently, it is to maximize the deep-semantic similarities between users and their target items and minimize those with irrelevant ones. Formally, it is equivalent to minimizing the following loss function:

$$\begin{aligned} L(\Theta) &= -\sum_{(u,i^*)} \log(Recom(u,i^*)) \\ &= -\sum_{(u,i^*)} [\log(e^{Sim(u,i^*)}) - \log(\sum_{i' \in I} e^{Sim(u,i')})], \end{aligned} \quad (8)$$

where Θ represents the parameters W_j and b_j in the neural network; (u, i^*) are training samples, which are pairs of a user u and his/her target item i^*, generated from assignments (u, t, i^*) in a training dataset.

In training, we first initialize the weight matrices W_j, using the random normal distribution, and initialize the biases b_j to be zero vectors; the model is then trained by back-propagation using mini-batch gradient descent; finally, the training stops when the model converges or reaches the maximum training runs. As for regularization, validation-based early stopping is used to avoid overfitting.

4.1 Negative Sampling

Although the loss function in Equation (8) is computable, it is computationally very expensive. This is because, for each training sample (u, i^*) in each training run, the second term requests to compute and sum the deep-semantic similarities between u and all candidate items in I. In the real-world context, the number of candidate items for an online recommendation system is usually very large (millions), and training a deep neural network often requires numerous training samples (millions) and many (thousands) training runs; the training of the DSPR model is thus very time-consuming in practice. However, this term is important: with its help, minimizing the loss function not only maximizes the deep-semantic similarities between the given user and his/her target items, but also minimizes those with irrelevant items. Consequently, it helps to distinguish the target item from the irrelevant ones.

To tackle this dilemma, we use *negative sampling* [7] to greatly reduce the time needed to process each training sample, and to ensure the scalability of DSPR. In negative sampling, instead of using all irrelevant items, for each training sample, we randomly sample only a small portion of irrelevant items from the set of candidate items as *negative examples* to approximate the noise and to differentiate target items from irrelevant ones. Consequently, the negative-sampling-based loss function is formally defined as follows:

$$L^{NS}(\Theta) = -\sum_{(u,i^*)} [\log(e^{Sim(u,i^*)}) - \log(\sum_{(u,i^-) \in D^-} e^{Sim(u,i^-)})],$$

where (u, i^-) are negative samples, which are contained in a negative dataset D^- and generated by randomly sampling S negative examples i^- for each training sample (u, i^*).

Table 1: Dataset Information

Users (u)	Tags (t)	Items (i)	Assignments ((u, t, i^*))
1 843	3 508	65 877	339 744

5. EXPERIMENTS

To show the strength of the DSPR with negative sampling (DSPR-NS) model in solving the uncontrolled vocabulary problem and offering superior personalized recommendation performance, we use three models based on state-of-the-art solutions, clustering [8] and autoencoders [10], as the baselines: (i) Clustering-based cosine similarity (**CCS**): hierarchical clustering [8] is used to model the users and items as cluster-based feature vectors, upon which content-based filtering using cosine similarity is applied for recommendations. (ii) Clustering-based collaborative filtering (**CCF**): CCF is similar to CCS but applies user-based collaborative filtering for recommendations. (iii) Autoencoder-based collaborative filtering (**ACF**) [10]: an autoencoder is used to obtain abstract representations of user profiles, upon which collaborative filtering is applied for recommendations.

The experiments are performed on the same public real-world dataset as used in [10], which is gathered from the Delicious bookmarking system and released in HetRec 2011 [3]. After using the same pre-processing to remove the infrequent tags that are used less than 15 times, the resulting dataset is as shown in Table 1. As we assume that the target items of a given user are the ones annotated by this user, we randomly select 80% of the assignment data as training set, 5% as validation set, and 15% as test set. The assignments (u, t, i^*) in the training set are used to construct user and item profiles and to extract the user-item pairs (u, i^*) as training samples. We also extract user-item pairs from the assignments in the validation set as validation samples, which are used to avoid over-fitting by early stopping. Finally, user-item pairs extracted from the assignments in the test set are used as test samples to evaluate the recommendation performance. Specifically, for each test sample (u, i^*), we first pass the profiles of u and all candidate items in I through the well-trained neural networks; then the relevance between u and all candidate items are computed based on Equation (7); finally, a recommendation list is generated where all candidate items are ranked according to their relevancy to u.

All models are implemented using Python and Theano and run on a GPU server with an NVIDIA Tesla K20 GPU and 8GB GPU memory. The parameters of DSPR-NS are set as follows: (i) # of hidden layers (i.e., K): 3; (ii) # of neurons in the first, second, and third hidden layers: 3 000, 300, and 128, respectively; (iii) training batch size: 128; (iv) # of negative examples for each training sample (i.e., S): 127; (v) # of maximum training runs: 10 000; and (vi) learning rate for model training: 0.001.

Since users usually only browse the topmost recommended items, we apply the *precision at k (P@k)*, *recall at k (R@k)*, and *F1-score at k (F@k)* as evaluation metrics. To take into account the order of items, we also employ the *mean average precision (MAP)* and *mean reciprocal rank (MRR)* as metrics to give greater importance to items ranked higher.

5.1 Main Results

Table 2 shows in detail the personalized recommendation performance of DSPR-NS and the three baselines in terms of *P@k*, *R@k*, *F@k*, MAP, and MRR, where four cut-off ranks $k = 1$, 5, 10, and 20 are selected.

Table 2: Recommendation Performance of Different Models (in %)

Models	$P@1$	$P@5$	$P@10$	$P@20$	$R@1$	$R@5$	$R@10$	$R@20$	$F@1$	$F@5$	$F@10$	$F@20$	MAP	MRR
CCF	1.194	0.868	0.814	0.667	0.089	0.417	0.765	1.158	0.165	0.564	0.789	0.847	0.416	0.200
ACF	1.465	1.139	0.950	0.798	0.194	0.561	0.891	1.370	0.342	0.752	0.919	1.008	0.606	0.252
CCS	3.527	2.279	1.970	1.712	0.263	0.892	1.637	2.741	0.490	1.282	1.788	2.108	1.254	0.523
DSPR-NS	**22.68**	**17.41**	**14.86**	**11.43**	**1.750**	**5.772**	**9.219**	**13.21**	**3.250**	**8.669**	**11.38**	**12.26**	**8.097**	**3.517**

Table 3: Training Efficiency and Effectiveness

	DSPR-O	DSPR-NS		
# of runs	500	500	1 000	10 000
time (hrs)	124.0	1.121	2.243	22.42
MRR-VS	0.02172	0.02157	0.02367	0.03555

The results in Table 2 show that our DSPR-NS model achieves the best personalized recommendation performance. It significantly outperforms the three other baselines in all metrics; e.g., the MRR of DSPR-NS is roughly 5.7, 13, and 16.5 times better than the ones of CCS, ACF, and CCF, respectively; also, the comparison results in $P@k$, $R@k$, $F@k$, and MAP are similar. The superior performance of DSPR-NS mainly has the following two reasons: (i) the training objective of DSPR-NS is directly correlated with distinguishing the user's target items from the irrelevant ones; so, the resulting abstract features for user and item profiles are much more effective representations for personalized recommendation than those generated by clustering and autoencoders; (ii) the relevance between users and items in DSPR-NS is directly computed using the deep-semantic similarities between user and item profiles; so, it can achieve more accurate recommendations than CCF and ACF, where the recommendations are only based on user profiles, so the relevance has to be inferred indirectly from other similar users.

5.2 Efficiency and Scalability

To investigate the training efficiency and scalability of DSPR, besides DSPR with negative sampling (DSPR-NS), we construct another model (DSPR-O) which is identical to DSPR-NS, except using the original loss function in Equation (8) for model training. The training time is recorded to compare the training efficiency of two models. But the standard training loss is not suitable for evaluating training effectiveness, as DSPR-O and DSPR-NS use different loss functions. Here, we use MRR on validation samples (MRR-VS) to measure the training quality, it is because (i) both training objectives are to get better personalized recommendations, so the higher the MRR-VS the better the model; (ii) MRR-VS is monitored during the training process to avoid over-fitting, so using it will not increase the training time.

As shown in Table 3, it costs DSPR-O 124.0 hours to finish the training of 500 runs, but DSPR-NS only takes 1.121 hours (i.e., 109.6 times quicker); meanwhile, the MRR-VS of two models are almost the same (0.02172 vs. 0.02157). This indicates that negative sampling can significantly enhance the training efficiency of DSPR, while maintaining almost the same training effectiveness. In addition, training 500 runs is usually not enough; more runs are required to train a good recommendation model. As shown in Table 3, it takes only 22.42 hours for DSPR-NS to finish the training of 10 000 runs and enhance MRR-VS up to 0.03555. However, without negative sampling, such training will cost DSPR-O 103.3 days, which is very impractical. So negative sampling greatly increases the scalability of DSPR and makes it possible to properly train DSPR in real-world situations.

6. SUMMARY AND OUTLOOK

We have proposed a tag-aware personalized recommendation system using a deep-semantic similarity model, DSPR, to extract recommendation-oriented representations for social tags, to address the uncontrolled vocabulary problem and to achieve superior personalized recommendations. We have also proposed to use negative sampling to greatly reduce the system's training time and to ensure a good scalability in practice. Experiments show that DSPR significantly outperforms the state-of-the-art baselines in personalized recommendation in terms of all selected metrics.

In the future, more extensive experiments will be conducted to further evaluate the performance of DSPR on different parameter settings and additional folksonomy datasets, e.g., Last.fm. Also, we plan to use hybrid learning signals or more sophisticated neural networks, e.g., convolutional network or LSTM, to learn more effective abstract tag representations for tag-aware personalized recommendations.

Acknowledgments

This work was partially supported by the UK EPSRC grants EP/J008346/1, EP/L012138/1, and EP/M025268/1. We acknowledge the use of Oxford University's ARC resources.

7. REFERENCES

[1] M. R. Bouadjenek, H. Hacid, M. Bouzeghoub, and A. Vakali. Using social annotations to enhance document representation for personalized search. In *SIGIR*, 2013.

[2] I. Cantador, A. Bellogín, and D. Vallet. Content-based recommendation in social tagging systems. In *RecSys*, 2010.

[3] I. Cantador, P. Brusilovsky, and T. Kuflik. Second Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec2011). In *RecSys*, 2011.

[4] A. M. Elkahky, Y. Song, and X. He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *WWW*, 2015.

[5] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information retrieval in folksonomies: Search and ranking. In *ESWC*, 2006.

[6] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*, 2013.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[8] A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *RecSys*, 2008.

[9] K. H. Tso-Sutter, L. B. Marinho, and L. Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *SAC*, 2008.

[10] Y. Zuo, J. Zeng, M. Gong, and L. Jiao. Tag-aware recommender systems based on deep neural networks. *Neurocomputing*, 2016.