Tag-Aware Personalized Recommendation Using a Hybrid Deep Model

Zhenghua Xu[1]*, Thomas Lukasiewicz[1], Cheng Chen[2]†, Yishu Miao[1], Xiangwu Meng[2]
[1]Department of Computer Science, University of Oxford, United Kingdom
[2]School of Computer Science, Beijing University of Posts and Telecommunications, China
{zhenghua.xu, thomas.lukasiewicz, yishu.miao}@cs.ox.ac.uk, {ccbupt, mengxw}@bupt.edu.cn

Abstract

Recently, many efforts have been put into 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 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). To ensure the scalability in practice, we further propose to improve this model's training efficiency by using hybrid deep learning and negative sampling. Experimental results show that our approach can significantly outperform the state-of-the-art baselines in tag-aware personalized recommendation (3.8 times better than the best baseline), and that using hybrid deep learning and negative sampling can dramatically enhance the model's training efficiency (hundreds of times quicker), while maintaining similar (and sometimes even better) training quality and recommendation performance.

1 Introduction

In the era of the 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* [Hotho *et al.*, 2006]). Since social tags are good summaries of the relevant items and the users' preferences, and they 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* [Cantador *et al.*, 2010; Shepitsen *et al.*, 2008] or *collaborative filtering* [Bouadjenek *et al.*, 2013; Tso-Sutter *et al.*, 2008]. 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 into very sparse, redundant, and ambiguous tag information, which greatly degrades the performance of tag-aware recommendation systems.

A solution to this problem is to apply clustering in the tag space [Shepitsen *et al.*, 2008]; however, clustering requests to compute the similarity between tags, which is usually very time-consuming. Another solution is to use autoencoders; in [Zuo *et al.*, 2016], abstract feature representations for tag-based user profiles are first modeled by autoencoders and then used as inputs of user-based collaborative filtering to generate recommendations. Although this method is reported to achieve better performance than the clustering-based collaborative filtering method [Zuo *et al.*, 2016], it still suffers from the following drawback: the model's learning signal is not directly related to the objective of personalized recommendation, i.e., distinguishing the user's target items from the irrelevant ones; so, the resulting abstract feature representations may not be effective for personalized recommendation.

In this paper, motivated by the above observations, we propose to address the uncontrolled vocabulary problem by using deep neural networks to map 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 the *deep-semantic similarity-based personalized recommendation* (DSPR) model. DSPR has the following advantages: (i) the deep model is trained using a recommendation-oriented learning signal, which 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) Deep neural networks in DSPR extract more abstract and denser features layer-by-layer, so DSPR overcomes sparsity and redundancy problems in the input tag space. (iii) Also, the input synonyms (resp., homonyms) have similar (resp., different) impact on output features of DSPR, which addresses the ambiguity problem in the social tag space. (iv) Interpretability (transparency) is a common concern for some deep learning applications; this problem can be remedied in DSPR by finding the most influential input tags for each abstract feature in the output deep feature space and then using them to summarize the semantics of corresponding features.

Despite achieving superior performance, the training of

*Principal corresponding author

†Corresponding author

the DSPR model is usually very time-consuming in practice, mainly because DSPR has many hidden layers and a huge number of candidate items. Specifically, on the one hand, since the deep neural networks in DSPR have many hidden layers, when learning signals are backpropagated to the first few layers, they become minuscule and insignificant, resulting into numerous training runs needed to reach model convergence (widely known as *diffusion of gradients*). On the other hand, 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. Since the number of candidate items for an online recommendation system is usually very large (millions), and training deep neural networks often requires many training samples, the processing of each training run is computationally very expensive.

To solve the former problem, we propose an approach, called *hybrid deep learning*, which directly integrates autoencoders with the neural networks of DSPR to generate additional learning signals based on reconstruction errors. Experiments show that hybrid deep learning can greatly accelerate the deep model's learning progress and reduce training runs needed for model convergence by tens of times. As for the latter problem, to reduce the processing time needed for each training run, we further use *negative sampling* [Mikolov *et al.*, 2013] to randomly sample a small number of negative examples to approximate the noise. The resulting efficient model is called *hybrid deep learning-based personalized recommendation with negative sampling* (HDLPR-NS).

The contributions of this paper are briefly as follows: (i) We propose DSPR which uses deep neural networks to solve the uncontrolled vocabulary problem in social tags. (ii) We further propose HDLPR-NS, which uses hybrid deep learning and negative sampling to achieve very efficient model training. (iii) Experimental results show that DSPR and HDLPR-NS both significantly outperform the state-of-the-art baselines in tag-aware personalized recommendation (3.8 times better than the best baseline), and that, by using hybrid deep learning and negative sampling, the model training of HDLPR-NS is hundreds of times more efficient than DSPR, while achieving similar (and sometimes even better) training quality and recommendation performance.

2 Related Work

Deep learning has already been successfully applied in many search and recommendation applications, such as music [Van den Oord *et al.*, 2013], movie [Salakhutdinov *et al.*, 2007], and item recommendation [Elkahky *et al.*, 2015], and Web search [Huang *et al.*, 2013]. Similarly to our work, Elkahky *et al.* [2015] and Huang *et al.* [2013] use deep-semantic similarity models with a ranking-oriented training objective. But these models are very different from the ones proposed here: (i) they are not tag-aware systems and not designed to solve the redundancy, sparsity, and ambiguity problems in the tag space; (ii) hybrid deep learning is not applied in these works to accelerate the convergence of the model; (iii) instead of using negative sampling, these two works intentionally assume the number of candidate items to be very small (5 in [Huang *et al.*, 2013] and 10 in [Elkahky *et al.*, 2015]) to make the model trainable. Clearly, this assumption is unreasonable in real-world situations. Negative sampling has been introduced first in the NLP community to learn word representations more efficiently [Mikolov *et al.*, 2013]. To our knowledge, we are the first who apply hybrid deep learning and negative sampling to enhance the training efficiency of deep-semantic similarity-based recommendation or search.

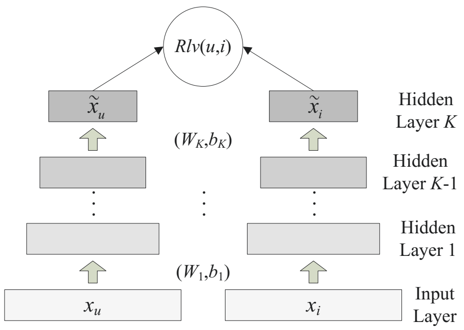


Figure 1: Overview of DSPR

A widely used solution for the diffusion of gradients is to pre-train each layer before using back-propagation to fine-tune the entire deep neural network [Erhan *et al.*, 2010], where the pre-training is usually done by restricted Boltzmann machines (RBMs) [Hinton and Salakhutdinov, 2006; Hinton *et al.*, 2006] or autoencoders [Bengio *et al.*, 2007]. Instead of using autoencoders for pre-training, in this work, we propose to overcome the bottleneck of back-propagation by directly integrating autoencoders into the network to generate additional learning signals based on reconstruction errors. Here, hybrid deep learning is used individually to accelerate the deep model's learning progress; but it can also be used as a complement of pre-training to speed up the learning process in the "fine-tuning" step.

3 Preliminaries

Personalized recommendation is defined as follows. Given a user u and a set of items $\{i_1, \ldots, i_n\}$, the system produces a ranked recommendation list $\tau = [i_1 \geq i_2 \geq \cdots \geq i_n]$ s.t. $i_a \geq i_b$ iff $Rlv(u, i_a) \geq Rlv(u, i_b)$, where $Rlv(u, i)$ is a function measuring how relevant an item i is to the user u.

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 [Hotho *et al.*, 2006].

A *user profile* is a feature vector $x_u = (g^u_1, \ldots, g^u_M)$, where $M = |T|$ is the tag vocabulary's size, and $g^u_j = |\{(u, t_j, i) \in A \mid i \in I\}|$ is the number of times that user u annotates items with tag t_j. Similarly, an *item profile* is a vector $x_i = (g^i_1, \ldots, g^i_M)$, where $g^i_j = |\{(u, t_j, i) \in A \mid u \in U\}|$ is the number of times that the item i is annotated with the tag t_j [Cantador *et al.*, 2010].

4 Deep-semantic Similarity-based Personalized Recommendation

Figure 1 shows an overview of the *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 above) 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, ranked recommendation lists are generated using relevance scores, computed by applying the softmax function on the resulting similarities.

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, respectively, for the jth hidden layer, and K is the total number of hidden layers. The outputs of the 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

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

and called *deep-semantic similarity*.

Finally, the relevance scores [Huang *et al.*, 2013] of items i to a given user u are measured by applying the softmax function on the resulting deep-semantic similarities between u and i, which are then used to rank a personalized recommendation list for the given user u. Formally,

$$Rlv(u,i) = e^{Sim(u,i)} \,/\, \textstyle\sum_{i' \in I} e^{Sim(u,i')}. \tag{7}$$

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

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

where Θ represents the parameters W_j and b_j in the neural networks; (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.

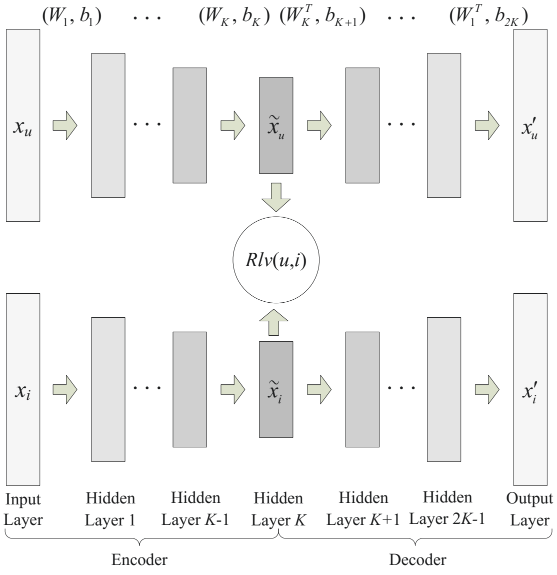


Figure 2: Overview of HDLPR

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.

5 Hybrid Deep Learning-based Personalized Recommendation with Negative Sampling

To enhance DSPR's training efficiency and ensure the scalability in practice, we propose to use *hybrid deep learning* and *negative sampling* [Mikolov *et al.*, 2013] to reduce the number of training runs needed for model convergence and to reduce the processing time needed for each training run. This is called *hybrid deep learning-based personalized recommendation with negative sampling* (HDLPR-NS).

5.1 Hybrid Deep Learning

Training deep neural networks using back-propagation is difficult, because the learning signals become minuscule and insignificant when they are backpropagated to the first few layers, which results into a very slow learning progress and numerous training runs for model convergence. This problem is called *diffusion of gradients*, which can be remedied by pre-training each layer before using back-propagation to fine-tune the entire network [Hinton and Salakhutdinov, 2006; Bengio *et al.*, 2007]. In this work, we propose a new solution called *hybrid deep learning* with an idea of integrating autoencoders with the neural networks of DSPR to generate additional learning signals based on *reconstruction errors*, which is then combined with the deep-semantic similarity-based relevance scores in DSPR to form a hybrid deep learning signal for model training. The model is called *hybrid deep learning-based personalized recommendation (HDLPR)*.

Intuitively, for the following reason, adding reconstruction errors as learning signals can greatly accelerate the deep model's learning progress and reduce training runs needed for model convergence: In DSPR, the deep-semantic similarity-based learning signal becomes very weak when it is back-propagated to the first few layers, so learning the first few weight matrices, e.g., W_1 and W_2, is very slow. In HDLPR, since we use *tied weights* in autoencoders, i.e., the weight matrices in the decoder are the transposes of those in the encoder, the reconstruction-error-based learning signal will be used to first update W_1^T, then back-propagated to updating W_2^T, W_3^T, and so on. As updating W_j^T is equivalent to updating W_j, it remedies the diffusion of gradients in DSPR.

Figure 2 shows the overall process of the proposed hybrid deep learning-based personalized recommendation (HDLPR) model. The structure of HDLPR is similar to the one of DSPR, but adds K layers to form a decoder; so, by taking the first $K+1$ layers as encoder, we convert each neural network to an autoencoder with tied weights. Then, the decoders take the abstract feature representations of user and item profiles, $\widetilde{x}_u$ and $\widetilde{x}_i$, as inputs and generate reconstructed user and item profiles in their output layers, denoted x'_u and x'_i. Finally, the reconstruction errors of user (resp., item) profiles are computed as the Euclidean (i.e., L2) norms of the differences between x_u (resp., x_i) and x'_u (resp., x'_i).

Formally, the definitions of layers in encoders are the same as the ones in the neural networks of DSPR. As for the decoders, the intermediate output of the $K+j$th hidden layer h_{K+j}, $j \in \{1, \dots, K-1\}$, is formally defined as:

$$h_{K+j}(u) = \tanh(W_{K-(j-1)}^T h_{K+(j-1)}(u) + b_{K+j}), \tag{9}$$

$$h_{K+j}(i) = \tanh(W_{K-(j-1)}^T h_{K+(j-1)}(i) + b_{K+j}), \tag{10}$$

where $W_{K-(j-1)}^T$ is the transpose of $W_{K-(j-1)}$, and b_{K+j} is the bias vector for the $(K+j)$th hidden layer. The outputs of the $(2K-1)$th hidden layer are used to generate reconstructed user and item profiles, denoted x'_u and x'_i, in the output layer:

$$x'_u = \tanh(W_1^T h_{2K-1}(u) + b_{2K}), \tag{11}$$

$$x'_i = \tanh(W_1^T h_{2K-1}(i) + b_{2K}), \tag{12}$$

Then, the reconstruction errors of user (resp.,item) profiles are computed as the Euclidean (i.e., L2) norms of the differences between x_u (resp., x_i) and x'_u (resp., x'_i). By integrating the reconstruction errors with the deep-semantic similarity-based relevance scores in DSPR, the training objective of HDLPR is to maximize the relevance scores of target items and also to minimize the reconstruction errors of the user and target item profiles in each training sample. Formally, it is equivalent to minimizing the following hybrid loss function:

$$\begin{aligned} L_h(\Theta) = {} & \lambda_\theta(\textstyle\sum_{j=1}^{K}\|W_j\|^2 + \sum_{j=1}^{2K}\|b_j\|^2) \\ & + \lambda_e \textstyle\sum_{(u,i^*)}(\|x'_u - x_u\| + \|x'_{i^*} - x_{i^*}\|) \\ & - (1-\lambda_\theta-\lambda_e)\textstyle\sum_{(u,i^*)}[\log(e^{Sim(u,i^*)}) \\ & - \log(\textstyle\sum_{i' \in I} e^{Sim(u,i')})], \end{aligned} \tag{13}$$

where the first term is a L2 regularization used to prevent overfitting; the second term is the sum of reconstruction errors of the user and target item profiles in each training sample; and the third term is the deep-semantic similarity-based learning signal, $L(\Theta)$, as defined in Eq. (8); λ_θ and λ_e are parameters representing different importances of the corresponding terms, where $\lambda_e > 0$, $\lambda_\theta > 0$, and $\lambda_\theta + \lambda_e < 1$.

5.2 Negative Sampling

Although hybrid deep learning greatly reduces the number of training runs needed for model convergence, the processing of each training run in both DSPR and HDLPR is still very time-consuming, since the deep-semantic similarity-based loss function $L(\Theta)$ as defined in Eq. (8), which is also the third term of $L_h(\Theta)$, is computationally very expensive.

Specifically, for each training sample (u, i^*) in each training run, the second term of $L(\Theta)$ requests to compute and sum the deep-semantic similarities between u and all candidate items in I. In practice, 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; therefore, the cost of processing each training run in both DSPR and HDLPR is very high. However, this term is essential: with its help, minimizing $L(\Theta)$ not only maximizes the deep-semantic similarity 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, in this work, we use *negative sampling* [Mikolov *et al.*, 2013] to greatly reduce the time needed to process each training sample. In negative sampling, instead of using all irrelevant items, for each training sample, we randomly sample only a small number (S) of irrelevant items from the set of candidate items as *negative examples* to approximate the noise and to differentiate target items from irrelevant ones. The resulting efficient model is called HDLPR with negative sampling (HDLPR-NS); and its loss function is formally defined as follows:

$$\begin{aligned} L_h^{NS}(\Theta) = {} & \lambda_\theta(\textstyle\sum_{j=1}^{K}\|W_j\|^2 + \sum_{j=1}^{2K}\|b_j\|^2) \\ & + \lambda_e \textstyle\sum_{(u,i^*)}(\|x'_u - x_u\| + \|x'_{i^*} - x_{i^*}\|) \\ & - (1-\lambda_\theta-\lambda_e)\textstyle\sum_{(u,i^*)}[\log(e^{Sim(u,i^*)}) \\ & - \log(\textstyle\sum_{(u,i^-) \in D^-} e^{Sim(u,i^-)})], \end{aligned} \tag{14}$$

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^*).

6 Experiments

Three state-of-the-art solutions for the uncontrolled vocabulary problem as baselines are: (i) Clustering-based cosine similarity (**CCS**): hierarchical clustering [Shepitsen *et al.*, 2008] models the users and items as cluster-based feature vectors, upon which content-based filtering with the cosine similarity is used for recommendations. (ii) Clustering-based collaborative filtering (**CCF**): CCF is similar to CCS in feature modeling, but uses user-based collaborative filtering for recommendations. (iii) Autoencoder-based collaborative filtering (**ACF**) [Zuo *et al.*, 2016]: an autoencoder is used to obtain abstract representations of user profiles, on which user-based collaborative filtering is applied for recommendations.

For a fair comparison, the experiments are performed on the same public real-world datasets, Delicious and Last.Fm, as used in [Zuo *et al.*, 2016], which are gathered from the Delicious bookmarking system and the Last.Fm online music system, respectively, and are both released in HetRec

Table 1: Dataset information

Dataset	Users	Tags	Items	Assignments
Delicious	1 843	3 508	65 877	339 744
Last.Fm	1 808	2 305	12 212	175 641

2011 [Cantador *et al.*, 2011]. After using the same preprocessing to remove infrequent tags, which are used less than 15 (resp., 5) times in Delicious (resp., Last.Fm), the information of the resulting datasets is shown in Table 1.

As we assume that the target items of a given user are those annotated by this user, for both datasets, 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.

All models are implemented using Python and Theano, and run on a GPU server [Richards, 2015] with an NVIDIA Tesla K40 GPU and 12GB GPU memory. The parameters of DSPR are selected by grid search and set as follows: (i) # of hidden layers (i.e., K) is 3; (ii) # of neurons in the 1st, 2nd, and 3rd hidden layers are 2 000, 300, and 128, respectively; (iii) training batch size is 128; and (iv) learning rate for model training is 0.005. These parameters are the same in HDLPR-NS, with the following additional settings: (v) two extra hidden layers, and # of neurons in the 4th and 5th hidden layers are 300 and 2 000, respectively; (vi) the balancing parameters λ_θ and λ_e are set to 0.01 and 0.2, respectively; and (vii) # of negative examples for each training sample (i.e., S) is 127.

The most popular metrics for the evaluation of recommendation systems are precision, recall, and F1-score [Bobadilla *et al.*, 2013]. Since users usually only browse the topmost recommended items, we apply these metrics at a given cut-off rank k, i.e., considering only the top-k results on the recommendation list, called *precision at k ($P@k$)*, *recall at k ($R@k$)*, and *$F1$-score at k ($F@k$)*. Since users always prefer to have their target items ranked in the front of the recommendation list, we also employ *mean reciprocal rank (MRR)* [Voorhees, 1999] as the evaluation metric, which gives greater importance to items ranked higher. The p-value [Rice, 1989] is used to measure the significance of improvements.

6.1 Main Results

Figure 3 depicts in detail the personalized recommendation performances of DSPR, HDLPR-NS, and the three baselines on the Delicious and Last.Fm datasets in terms of MRR, $P@k$, $R@k$, and $F@k$, where ten cut-off ranks $k = 5, 10, \ldots, 50$ are selected for evaluation.

In Figure 3, despite some slight differences due to the random weights' initializations and dataset partitions, the results of CCF and ACF in Figure 3 are highly consistent with the results reported in [Zuo *et al.*, 2016] in terms of orders of magnitude, tendencies, and relative performances. Furthermore, the performances of the proposed DSPR and HDLPR-NS models are very similar (e.g., p-value $= 94.31\%$ in MRR) on the Delicious dataset; while HDLPR-NS slightly outperforms DSPR on Last.Fm. This indicates that integrating hybrid deep learning and negative sampling with DSPR will not degrade its performance (and sometimes even improve it).

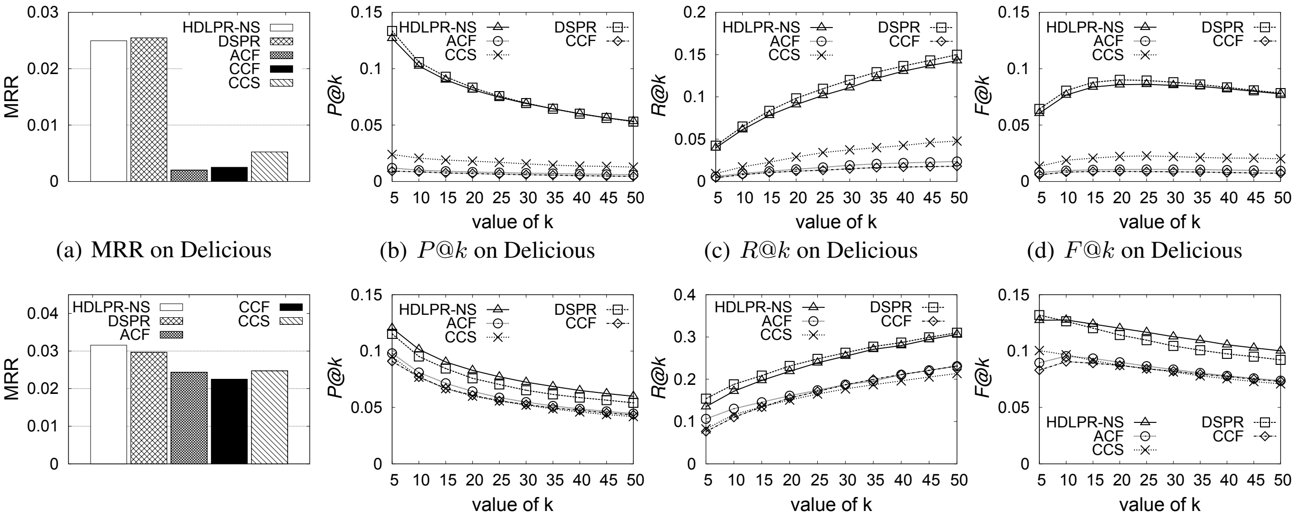
In general, Figure 3 shows that the DSPR and HDLPR-NS models both significantly outperform the three baselines in all metrics; e.g., the MRRs (resp., average $R@k$) of DSPR and HDLPR-NS are both more than 3.8 times (resp., 34%) better than the best baseline, CCS, on Delicious (resp., Last.Fm) with p-value both smaller than 0.04% (resp., 0.09%), and the improvements in other metrics are also similar. The superior performances of DSPR and HDLPR-NS are mainly because the training objectives of DSPR and HDLPR-NS are both 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 solely by clustering and autoencoders.

We also note in Figure 3 that DSPR and HDLPR-NS achieve much higher improvements on Delicious than on Last.Fm, as DSPR and HDLPR-NS maintain a relatively stable performance on both datasets, but the performance of the baselines dramatically degrades on Delicious: all metrics are $1/5$ or less than those on Last.Fm. This may be because: items in Last.Fm are solely in the domain of music, but items in Delicious are webpages in various domains; so, even with higher tag-removing threshold, Delicious still has a more variable, redundant, and ambiguous tag space than Last.Fm, which can not be properly handled by the baselines. This shows that DSPR and HDLPR-NS can solve the uncontrolled vocabulary problem better than the baselines; and the more uncontrolled the dataset, the higher the achieved improvement.

6.2 Efficiency and Scalability

We also investigate the training efficiency and scalability of DSPR and HDLPR-NS. The training time is recorded to compare the training efficiency of the two models. But the standard training loss is not suitable to compare the models' training quality, as DSPR and HDLPR-NS have different loss functions. Here, we use MRR on validation samples (MRR-VS) to measure the training quality, because (i) the training objectives of both models are to get a better performance in personalized recommendation, so the higher the MRR-VS, the better the models, and (ii) the values of MRR-VS are computed every 10 training runs to avoid over-fitting, so using it will not increase the training time.

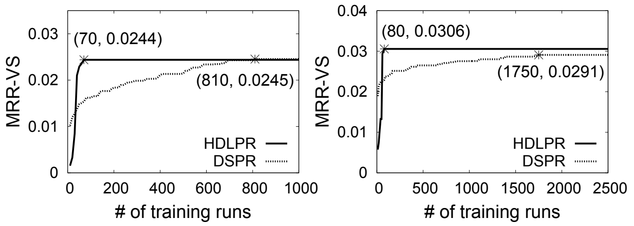
We first explore the effect of using hybrid deep learning. To avoid interference, we use HDLPR (hybrid model without negative sampling) for comparison. As shown in Figure 4, with the increase of the number of training runs, HDLPR converges much faster than DSPR on both datasets. DSPR takes 810 (resp., 1750) training runs to reach model convergence on Delicious (resp., Last.Fm), which costs HDLPR only 70 (resp., 80) runs. This shows that hybrid deep learning can significantly enhance the model's learning progress and reduce the training runs needed for model convergence by tens of times. Also, when the models converge, the MRR-VS of HDLPR is almost the same as that of DSPR on Delicious



(a) MRR on Delicious (b) $P@k$ on Delicious (c) $R@k$ on Delicious (d) $F@k$ on Delicious

(e) MRR on Last.Fm (f) $P@k$ on Last.Fm (g) $R@k$ on Last.Fm (h) $F@k$ on Last.Fm

Figure 3: The personalized recommendation performances in terms of $P@k$, $R@k$, $F@k$, and MRR



(a) Delicious (b) Last.Fm

Figure 4: Training progress in DSPR and HDLPR

(0.0244 in HDLPR vs. 0.0245 in DSPR) and is slightly higher on Last.Fm (0.0306 in HDLPR vs. 0.0291 in DSPR), which demonstrates that using hybrid deep learning will not degrade the models' training quality.

We then apply negative sampling on HDLPR and find that the resulting HDLPR-NS has a very similar training progress to HDLPR: as shown in Table 2, it takes HDLPR-NS 70 (resp., 80) runs to converge at MRR-VS $= 0.0243$ (resp., $= 0.0304$) on Delicious (resp., Last.Fm); these values are almost the same as those of HDLPR as shown in Figure 4. Therefore, using negative sampling will not influence the benefit of using hybrid deep learning. Furthermore, Table 2 also shows that, with the help of negative sampling, the time needed to process one training run on Delicious (resp., Last.Fm) is greatly reduced from 0.405 (resp., 0.1133) hours in DSPR to 0.0095 (resp., 0.0037) hours in HDLPR-NS. This proves that using negative sampling can substantially reduce the processing time needed for each training run.

Consequently, the total time-cost to reach model convergence in HDLPR-NS is only 0.665 (resp., 0.296) hours on Delicious (resp., Last.Fm), which is roughly 492 (resp., 668) times quicker than that in DSPR. In summary, by using hybrid deep learning and negative sampling, the model training efficiency of HDLPR-NS is hundreds of times better than the one of DSPR, while maintaining a similar training quality.

Table 2: DSPR vs. HDLPR-NS in training time and quality

	Delicious		Last.Fm	
	DSPR	HDLPR-NS	DSPR	HDLPR-NS
runs to converge	810	70	1750	80
MRR-VS	0.0245	0.0243	0.0291	0.0304
time for 1 run (hr)	0.405	**0.0095**	0.1133	**0.0037**
total time (hr)	328.0	**0.665**	198.3	**0.296**

7 Summary and Outlook

We have proposed a deep-semantic similarity model, DSPR, to address the uncontrolled vocabulary problem and to achieve superior personalized recommendations. We have also proposed an improved model, HDLPR-NS, to use hybrid deep learning and negative sampling to greatly reduce the system's training time and to ensure a good scalability in practice. Experiments show that DSPR and HDLPR-NS both significantly outperform the state-of-the-art baselines in personalized recommendation in all metrics. In addition, the training efficiency of HDLPR-NS is hundreds of times better than the one of DSPR, while maintaining a similar training quality and performance in personalized recommendation.

In the future, we will further improve our models with more sophisticated neural networks, e.g., recurrent networks. We will also apply the hybrid deep model on other real-world applications and conduct extensive experiments to get more insights about its performance in various practical situations.

Acknowledgments

This work was supported by the UK EPSRC grants EP/J008346/1, EP/L012138/1, and EP/M025268/1, and by The Alan Turing Institute under the EPSRC grant EP/N510129/1. Cheng Chen and Xiangwu Meng are supported by the Mutual Project of Beijing Municipal Education Commission.

References

[Bengio *et al.*, 2007] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, volume 19, pages 153–160. 2007.

[Bobadilla *et al.*, 2013] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

[Bouadjenek *et al.*, 2013] Mohamed Reda Bouadjenek, Hakim Hacid, Mokrane Bouzeghoub, and Athena Vakali. Using social annotations to enhance document representation for personalized search. In *Proceedings of the International ACM SIGIR Conference*, pages 1049–1052, 2013.

[Cantador *et al.*, 2010] Iván Cantador, Alejandro Bellogín, and David Vallet. Content-based recommendation in social tagging systems. In *Proceedings of the ACM Conference on Recommender Systems*, pages 237–240, 2010.

[Cantador *et al.*, 2011] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. Second Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec2011). In *Proceedings of the ACM Conference on Recommender Systems*, pages 387–388, 2011.

[Elkahky *et al.*, 2015] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the International Conference on World Wide Web*, pages 278–288, 2015.

[Erhan *et al.*, 2010] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, 2010.

[Hinton and Salakhutdinov, 2006] Geoffrey E. Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[Hinton *et al.*, 2006] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[Hotho *et al.*, 2006] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information retrieval in folksonomies: Search and ranking. In *Proceedings of the European Semantic Web Conference*, pages 411–426, 2006.

[Huang *et al.*, 2013] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 2333–2338, 2013.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[Rice, 1989] William R. Rice. Analyzing tables of statistical tests. *Evolution*, 43(1):223–225, 1989.

[Richards, 2015] Andrew Richards. University of Oxford Advanced Research Computing. 2015. Zenodo.10.5281/zenodo.22558.

[Salakhutdinov *et al.*, 2007] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the International Conference on Machine learning*, pages 791–798, 2007.

[Shepitsen *et al.*, 2008] Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the ACM Conference on Recommender Systems*, pages 259–266, 2008.

[Tso-Sutter *et al.*, 2008] Karen H. L. Tso-Sutter, Leandro Balby Marinho, and Lars Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proceedings of the ACM Symposium on Applied Computing*, pages 1995–1999, 2008.

[Van den Oord *et al.*, 2013] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651, 2013.

[Voorhees, 1999] Ellen M. Voorhees. The TREC-8 question answering track report. In *TREC*, volume 99, pages 77–82, 1999.

[Zuo *et al.*, 2016] Yi Zuo, Jiulin Zeng, Maoguo Gong, and Licheng Jiao. Tag-aware recommender systems based on deep neural networks. *Neurocomputing*, 204:51–60, 2016.