2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)

Social Tag Embedding for the Recommendation with Sparse User-item Interactions

Deqing Yang*‡, Lihan Chen†, Jiaqing Liang†, Yanghua Xiao†§ and Wei Wang†
*School of Data Science, Fudan University, Shanghai, China. Email: yangdeqing@fudan.edu.cn
†School of Computer Science, Fudan University. Email: {lhchen16, l.j.q.light, shawyh, weiwang1}@fudan.edu.cn
§Shanghai Institute of Intelligent Electronics & Systems, Shanghai, China.

***Abstract*—Most of traditional recommender systems perform well only when sufficient user-item interactions, such as purchase records or ratings, have been obtained in advance, while suffering from poor performance in the scenario of sparse interactions. Addressing this problem, we propose a neural network based recommendation framework which is fed with user/item's original tags as well as the expanded tags from social context. Through embedding the latent correlations between tags into distributed feature representations, our model uncovers the implicit relationships between users and items sufficiently, exhibiting superior performance no matter whether sufficient user-item interactions are available or not. Furthermore, our framework can be further tailored for link prediction in networks, since recommending an item to a user can be recognized as predicting a link between them. The extensive experiments on two real recommendation tasks, i.e., Weibo followship recommendation and Douban movie recommendation, justify our framework's superiority to the state-of-the-art methods.**

***Index Terms*—recommendation, tag embedding, social context, user-item interactions**

I. Introduction

A. Background

Social media have experienced fast growth and attracted extensive research interests in the last decade. One of particular research interests on social media is *recommender system*, which has a wide range of real-world applications ranging from friend recommendation [6], product recommendation [33] to content recommendation [27]. Recommender system has become a critical component in almost all social media platforms. As the most popular recommendation scheme, *collaborative-filtering (CF for short) based* methods [12], [13] are ineffective when historical user-item interactions (e.g., purchase records or ratings) are not available or sparse. It is usually referred to as *cold start* or *data sparsity* problem. For another famous recommendation scheme, *content-based* methods [5], [1], users and items need to be characterized sufficiently by features, such as user activities, user/item descriptions or social relationships. However, profiling a user or item is quite complicated in general. A good user/item profiling approach should characterize a user with not only explicit features but also hidden or implicit features or factors [10]. Many traditional recommendation approaches employ these hidden factors poorly, thus have low recommendation performance.

This work is supported by National Key R&D Program of China (No. 2017YFC0803700), Chinese NSFC Project (No. U1636207), Shanghai STCSM's R&D Program under Grant (No. 16JC1420400).
‡Corresponding author.
IEEE/ACM ASONAM 2018, August 28-31, 2018, Barcelona, Spain
978-1-5386-6051-5/18/$31.00

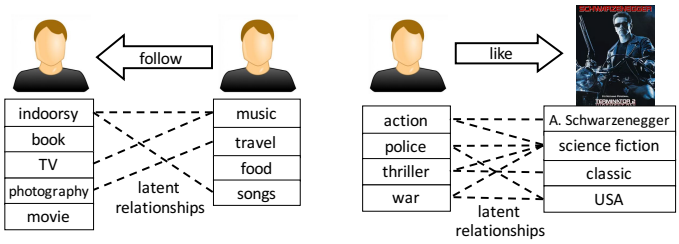
Recently, many deep learning (DL for short) based models have been developed to enhance recommendation performance [8] or alleviate the difficulty of data sparsity [26], [30]. Although these models are effective in employing latent features in data, there is still great space to improve these models. *First, some DL based solutions still suffer from sparse interaction data*. Most of these deep learning based models still follow CF scheme [9], [7], thus perform not well when only rare user-item interactions are available. Such interactions are very significant to infer user preferences and further to measure the matching degree between users and items. *Second, some DL based solutions rely on raw text data, leading to error-prone feature abstraction and selection*. Some content-based recommender systems use the DL models fed with free texts (or keywords), such as tweets or posts published in online forums [21], [28]. However, free texts in general are noisy and thus need many extra natural language processing to achieve a better performance. On the other hand, raw free texts are low-level features from which the high-level features, such as user preferences or intent, need to be extracted by complicated feature engineering.

In contrast, we notice that there is a recent trend that social platforms (such as Flickr, Facebook and Delicious) are encouraging users to publish tags for themselves or various resources on Web. The tags are often used to describe users' hobbies, professions or resources' topics, categories and etc. Therefore, besides user-item interactions, we can use tags to infer a user's preferences thus overcome the shortcoming of CF-based recommendation in the scenario of sparse user-item interactions. On the other hand, many tags can be directly used without extra refinement in real applications such as friend recommendation because tags are more clean and well-formed than purely free texts [32]. In this paper, we focus on *embedding tag latent relationships and social context to promote recommendation performance*.

B. Motivation

Our work is developed upon a simple observation that *latent semantic relationships of tags are critical for inferring social relationships between users or relationships between users and items*. We illustrate it with the examples in Fig. 1. Subfigure (a) displays that a Weibo[1] user follows another one who is labeled with completely different tags. Obviously, it is impossible to infer the followship between these two users just by the lexical similarity of their tag lists. However, we found that the latent semantic relationships (dashed lines)

[1]http://weibo.com. It is the most popular Chinese counterpart of Twitter and we translated all Chinese tags into English for better readability.



(a) Weibo's user-user relationship (b) Douban's user-movie relationship

Fig. 1. Examples of user-user/item relationship in two recommendation scenarios. (a) is an example of two Weibo users linked by a followship. (b) is an example of a Douban user and his/her favorite movie. Obviously, it is hard to identify the links (follow/like) just by the lexical similarity of their tag lists. However, the latent semantic relationships (dashed lines) between tags indicate the user-user/item relationships, motivating us to employ the latent semantic relationships between tags for recommendation.

inherent in the cooccurrence information of tag lists are good indicators of their social relationships. For example, in Weibo we found that *'indoorsy'*-*'music'* are co-used by 26,406 Weibo users, *'indoorsy'*-*'songs'* are co-used by 19,815 users and *'photography'*-*'travel'* are co-used by 15,977 users in our statistics dataset which contains 762,792 users in total. Such cooccurrence relationships reveal the latent semantic relationships between tags. More specifically, a person who is indoorsy is quite likely to like music or songs. In addition, many travelers are likely to use cameras (photography) to record everything he/she encounters during his/her trip. Sub-figure (b) shows a similar case on Douban[2] from which we infer that the user likes the movie by the latent semantic relationships on their tags instead of the lexical similarity between their tag lists. Specifically, the user tags specify the genres of his/her favorite movies. The movie tags tell us that many movies of science fiction are also the movies of action and thriller. Furthermore, A. Schwarzenegger often starred the movies of action. Therefore, using such latent relationships between tags is beneficial for accurate user/item profiling, thus promoting the performance of personalized recommendation.

C. Challenges and Contributions

We still have many issues to address before we effectively use the latent semantic relationships between tags. *First, how to reveal the latent semantic relationships between tags?* In general, we need an appropriate representation mechanism for tags so that the latent semantic relationships can be established. *Second, how to solve the tag sparsity issue?* The tags of users might be limited. Due to the privacy concern, some users are not necessarily willing to label themselves sufficiently. Thus, the tags for some users are sparse. Hence, we further need to enrich a user's description by expanding his/her tags.

For the first issue, we proposed a tag embedding model to generate distributed tag representations which encode the latent relationships among tags sophisticatedly. For the second issue, we expand a user's tag list with tags carefully selected from his/her social friends and incorporate these expanded tags into our embedding model. Finally, the tag representations are input as user/item's features into a neural classifier to implement personalized recommendation.

The contributions of this paper are summarized as follows:

[2]http://www.douban.com, it is a famous Chinese review website for movies, books, musics and etc.

1. We propose a representation learning model, namely *tag embedding*, to encode latent relationships between tags into distributed vectors, which is beneficial for user/item profiling.
2. Based on tag embedding model, we develop a neural recommendation framework fed with tag embeddings of users and items. Our framework achieves robust performance even when there exist only sparse user-item interactions. Moreover, our solution can be easily adapted to link prediction in networks since link prediction can be naturally modeled as a problem of relationship recommendation between objects.
3. We design and conduct extensive experiments on two real recommendation tasks. The results sufficiently justify that our framework significantly outperforms the state-of-the-art recommendation models and solves the targeted problems very well.

The rest of this paper is organized as follows. We introduce the details of our solution in Section 2. Section 3 and Section 4 are our experimental results and related work, respectively. At last, we conclude our work in Section 5.

II. Related Work

A. Word Embeddings

The Word Embeddings proposed by Mikolov et al. [19], [18] aim to find word representations which discover the relationships between a word to its surrounding words in sentences. The related techniques include the Skip-gram and continuous bag of words (CBOW), which can cast the learned linguistic regularities and semantics from text corpus into distributed word vectors. The authors in [3] claimed that word embeddings can be learned by latent semantic analysis (LSA), topic models and matrix factorization. Many literatures followed this popular and effective model in recent years, such as [15] in which the authors fused word topics into embeddings to enhance the performance of word similarity computing.

B. Embedding based Recommendation

Since word embeddings can well represent content-based features in semantics, the embedding model has been imported into some content-based recommender systems before. Many of previous works tried to embed the keywords extracted from documents instead of user tags, to enrich the content-based specification of users/items. The authors in [21] mapped the items to textual contents through embedding the keywords from Wikipedia documents, for their evaluated recommendation on MovieLens and DBbook datasets. [28] employed Word2Vec to capture the vector representations of tags to facilitate inductive matrix completion (IMC) which is used for recommending Tumblr blogs to follow. They also used the information of social networks as our work but their tags are indeed the keywords distilled from users' blog texts. In addition, some non-textual features can also be embedded to make effective recommendation, such as the user check-ins of Foursquare [22] and user search queries and clicked URLs [10]. As we claimed before, the keywords from free texts are generally noisy and thus not so effective as well-defined tags on the description of users/items.

C. Deep Learning based Recommendation

Besides embeddings features of user/items, there are a few literatures employing deep neural network to alleviate data sparsity problem in recommender systems. For example, [24] used Restricted Boltzmann Machines (RBM) on CF-based

movie recommendation. [26] proposed a novel AutoEncoder (AE) framework for CF. Based on the SDAE [29] of a deep neural network, [9] proposed Bayesian aSDAE to enhance CF-based recommendation performance. Furthermore, the authors of [30] proposed a hierarchical Bayesian model which jointly performs deep representation learning for content information and collaborative filtering for the ratings matrix. [4] utilized embedding technique to improve item-based CF. [8] strived to bridge the semantic gap in music by training deep convolutional neural networks to predict latent factors from music audio. Although these deep-based solutions were proved to be effective, a well formed user-item matrix needs being obtained as the prerequisite thus they are not practicable in the scenario of very spare user-item interactions. Comparatively, our solution achieves robust performance through uncovering latent relationships between tags which is not restricted by data sparsity as CF-based solutions. Although the authors in [14] also feed the information of social context into deep learning model, the objective of their work is comprehensive understanding user interests and behaviors rather than personalized recommendation. In addition, its cost in data collection and pre-processing is apparently large because the user features come from different social sources, which restricts its applications in real recommendation tasks.

III. Recommendation Solution

In this section, we introduce our neural network based framework for the recommendation of sparse user-item interactions. For a better presentation, we describe our framework w.r.t. a specific recommendation task, *Weibo followship recommendation*, although our model can be tailored for other similar recommendation task. In fact, a followship recommendation between two Weibo users can be regarded as predicting a link between two users in a social network. Accordingly, it is reasonable to model the task as a binary classification for two nodes in a network, i.e., there should (not) be a link between them. Therefore, we design a binary classification model which accepts user u and v as input and predict whether u will follow v.

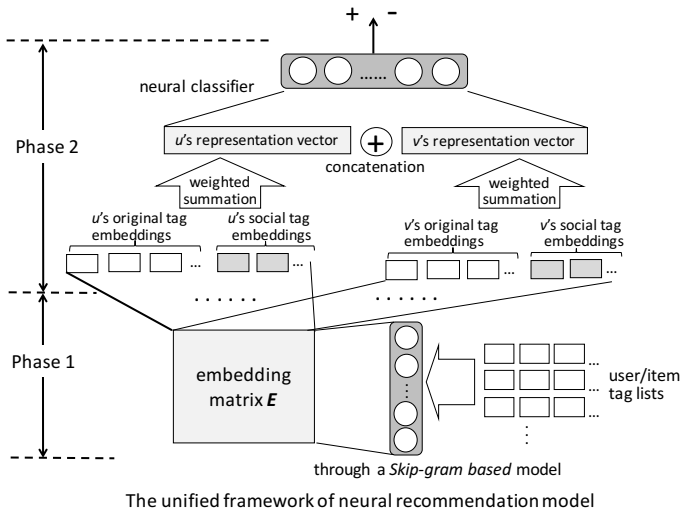


Fig. 2. The recommendation framework consisting of tag embedding model and neural classifier. The tag embeddings generated in Phase 1 constitute user representations, which are input into a neural classifier in Phase 2 to predict recommendation results.

The framework of our solution is illustrated in Fig. 2, consisting of two major phases:

Phase 1: Tag Embedding. It assembles a fully connected neural layer to project each one-hot tag representation to a latent feature vector (termed as *tag embedding* in this paper) in terms of tag cooccurrences.

Phase 2: Neural Classification. The tag embeddings generated in the first phase are input into a neural binary classifier, which decides whether a given user should follow another user as a friend or not. It is also applicable to determine whether an item deserves to be recommended to a user.

A. Tag Embedding Model

1) Model Representation: The similarity or correlation between tags is obviously a good indicator of the potential followships between Weibo users since tags well characterize users such as hobby, career, religion and etc. Therefore, our proposed model should find a good representation to fully reflect the similarity or correlation between tags. In this paper, we name such representation as *tag embedding*, which is effective in measuring semantic distance between tags, thus useful to measure the similarity between two users. We will show in our experiments that the similarity based on tag embeddings is superior to the metric based on tag lexicons. Next, we elaborate how to learn tag embeddings.

Inspired by the Skip-gram model [19], we use cooccurrence relationships between tags in each user's tag list to learn tag embeddings. The Skip-gram model is built to maximize the probability of a word conditioned on another word in the same sentence. The objective of the Skip-gram model is to learn the word representation vectors that are good at predicting the nearby words given the current word. According to this basic idea, given a user's tag set, denoted by $t_1, t_2, ..., t_S$, the objective of tag embedding model is to maximize the following conditional probability

$$\prod_{i=1}^{S} \prod_{1 \le j \le S \wedge j \ne i} p(t_j|t_i) \tag{1}$$

where $p(t_j|t_i)$ is the probability that tag t_j cooccurs with tag t_i given t_i. In the generic Skip-gram model, $p(t_j|t_i)$ is computed by the softmax function [20], but it is impractical to be learned due to computational complexity. Hence, we instead use *negative sampling* [19] to replace $p(t_j|t_i)$ by a Sigmoid function of the *embedding vectors* of t_i and t_j, which is denoted as $\sigma(\cdot)$. More formally, we first represent each tag as a one-hot vector $\boldsymbol{t}$ of N dimensions, where N is the total number of unique tags. And we also construct an *embedding matrix* $\boldsymbol{E} \in \mathbb{R}^{k \times N}$, also known as *lookup matrix*, where k is the dimension of embedding vectors in which each entry is a parameter of the model. Then, we can get a tag embedding vector through selecting one column from the matrix $\boldsymbol{E}$. Specifically, the embedding vector $\boldsymbol{e} \in \mathbb{R}^k$ is computed as

$$\boldsymbol{e} = \boldsymbol{E}\boldsymbol{t} \tag{2}$$

where $\boldsymbol{E}$ contains all parameters of the model. Accordingly, given two tag t_i and t_j, we get their embedding vectors $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ by Eq. 2. Then, we compute the probability that t_j cooccurs with t_i as the Sigmoid function of the inner product of $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$.

$$p(t_j|t_i) = \sigma(\boldsymbol{e}_j \cdot \boldsymbol{e}_i) = \frac{1}{1 + e^{-(\boldsymbol{e}_j \cdot \boldsymbol{e}_i)}} \tag{3}$$

Obviously, the similarity between $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ is proportional to their inner product. Thus, $p(t_j|t_i)$ is closer to 1 when $\boldsymbol{e}_j$ is closer to $\boldsymbol{e}_i$.

2) Model Learning: Next, we introduce how to train the tag embedding matrix $\boldsymbol{E}$. Specifically, a training sample is formalized as a triplet denoted by $< a, b, y >$, where a and b are two tags, and $y \in \{0, 1\} = L_{a,b}$, that is the class label of pair (a, b) indicating whether b cooccurs with a in a user's tag set. Thus, we have

$$L_{a,b} = \begin{cases} 1 & \text{if } b \text{ cooccurs with } a \text{ in a user's tag list} \\ 0 & \text{otherwise} \end{cases}$$

To generate positive samples, we collect each pair of tags that cooccur in any user's tag set. For any two tags which are never co-used by any user, we first use them to constitute a negative sample pool. Then, we randomly select enough samples from this pool as negative samples. In general, the number of negative samples is M times of the number of positive samples.

Next, we explain how to derive the objective function of embedding model. Given a tag a, we first use $C(a)$ to represent the tags that cooccur with tag a in someone's tag set. Then, we should maximize the following objective function

$$O_a = \prod_{b \in C(a) \wedge b \neq a} \left\{ \sigma(\boldsymbol{e}_b \cdot \boldsymbol{e}_a)^{L_{a,b}} \prod_{i=1}^{i=M} \mathbb{E}_{b_i \sim P} \left[1 - \sigma(\boldsymbol{r}_{b_i} \cdot \boldsymbol{r}_a)\right]^{1 - L_{a,b_i}} \right\} \tag{4}$$

where P is tag distribution on whole tag space, namely $\mathcal{T}$, from which a negative tag b_i ($b_i \notin C(a)$) is drew for M times. In our scenario, we compute P based on each tag's frequency at which it is used by Weibo users. This objective indicates that $\sigma(\boldsymbol{e}_b \cdot \boldsymbol{e}_a)$ should be as large as possible if $L_{a,b}$=1, or be as small as possible if $L_{a,b}$=0, which is proportional to the similarity between b and a. For $\mathcal{T}$, the global objective function is $\mathcal{O} = \prod_{a \in \mathcal{T}} O_a$. In general, we select negative logarithm of $\mathcal{O}$ as

$$\begin{aligned} \mathcal{L} &= -\log \prod_{a \in \mathcal{T}} O_a = -\sum_{a \in \mathcal{T}} \log O_a = -\sum_{a \in \mathcal{T}} \log \prod_{b \in C(a) \wedge b \neq a} \\ &\left\{ \sigma(\boldsymbol{e}_b \cdot \boldsymbol{e}_a)^{L_{a,b}} \prod_{i=1}^{i=M} \mathbb{E}_{b_i \sim P} \left[1 - \sigma(\boldsymbol{e}_{b_i} \cdot \boldsymbol{e}_a)\right]^{1 - L_{a,b_i}} \right\} \\ &= -\sum_{a \in \mathcal{T}} \sum_{b \in C(a) \wedge b \neq a} \\ &\left\{ \log \sigma(\boldsymbol{e}_b \cdot \boldsymbol{e}_a) + \sum_{i=1}^{i=M} \mathbb{E}_{b_i \sim P} \log \left[\sigma(-\boldsymbol{e}_{b_i} \cdot \boldsymbol{e}_a)\right] \right\} \end{aligned} \tag{5}$$

In order to learn the parameters in Eq. 5, i.e., $\boldsymbol{e}_a$, $\boldsymbol{e}_b$ and $\boldsymbol{e}_{b_i}$, we use *stochastic gradient descent* [11].

B. Neural Classification Model

Next, we elaborate our neural-network based classification model in Phase 2. It aggregates all features built in Phase 1 to generate recommendation results. Specifically, we use a binary classifier which predicts whether a user should follow another user based on their tag embeddings.

1) Model Description: To build such a binary classifier, we have to address two issues. First, we need to define the representation of a user. Second, we need a score function to quantify the likelihood that a user will follow another user.

Since we have already learned an embedding vector for each tag, we directly sum up the weighted embeddings of the tags that belong to a user's tag set. More formally, the representation vector of user u is derived by

$$\boldsymbol{r_u} = \sum_{i \in T_u} w_i \boldsymbol{e_i} \tag{6}$$

where T_u is u's tag set and w_i is the normalized weight of tag i, indicating the extent to which tag i characterizes user u. The operation in this step is similar to the pooling operation of CNN (Convolutional Neural Networks) [25], which is used to reduce the number of model parameters. But we do not use convolution to aggregate all tag embeddings because a user's tags have no order. We use the operation of normalized weighted summation rather than max/min/average since normalized weighted summation reserves more information built in the embeddings[3].

Next, we define the score function which indicates how likely user u will follow v. We first concatenate the representation vectors of u and v by $Concat(\boldsymbol{r_u}, \boldsymbol{r_v})$, which is further fed into a binary classifier. We use $Sigmoid$ function as the classifier function, thus the final prediction score (probability of followship) is

$$\hat{y}(u \to v) = \sigma\big(\boldsymbol{W} Concat(\boldsymbol{r_u}, \boldsymbol{r_v}) + b\big) \tag{7}$$

where b is a bias and $\boldsymbol{W}$ is a feature weight matrix of size $1 \times 2k$. The function can be viewed as output layer of the neutral network, in which $\boldsymbol{W}$ and b are the parameters learned through model training. Note that since $Concat(\boldsymbol{r_u}, \boldsymbol{r_v})$ is not symmetric, $\hat{y}(u \to v)$ is not necessarily equivalent to $\hat{y}(v \to u)$, which is consistent with the fact that Weibo followships have directions.

2) Model Training and Prediction: Like model training in the first phase, we also collect training samples represented by triplets $< u, v, y >$, where u and v are two users, and y indicates whether u follows v in the social network. That is, any two users that have followship relationship are used as a positive sample. We still build a negative sample pool by collecting two users without followship in random, and also use the objective function similar to Eq. 5 to learn the model parameters. The tag embeddings can be viewed as the hidden layer if we regard the whole framework combining Phase 1 and Phase 2 as a big neural network, as illustrated in Fig. 2.

When model parameters are learned, we decide the recommendation according to the value of $\hat{y}$. That is, we conclude that user v deserves to be followed by user u if $\hat{y}(u \to v)$ is larger than a threshold that is often set as 0.5 in binary classification. Alternatively, we could select top-n users with the largest $\hat{y}$ as the recommended followees to u. Furthermore, we can apply our framework to other scenarios of item recommendation if items are also profiled by theirs tags. It should be noted that our model can accomplish recommendation in real time because the embeddings of all tags are trained off-line in advance.

[3]It has been proved in our experiments but we do not list the results in this paper due to space limitation.

3) Embeddings Social Context: As mentioned before, many users' tags are not sufficient to profile themselves. Due to privacy concern, many users is not willing to label themselves by many tags or concrete tags that can characterize them richly. To address this issue, we expand the tag set of a user so that we can derive a more complete and precise tag description for the user.

According to the famous finding in social networks, i.e., the *homophily* [17], the characteristics of a user's friends can also indicate the characteristics of this user. Therefore, it motivates us to import the tags of a user's friends into the model to improve model performance because these new imported tags can bring more information to profile the objective user. We call such expanded tags as *social tags*. In this paper, we used the tag recommendation algorithm proposed in [31] to collect social tags including tag weights. This algorithm generates tags based on a recursive tag propagation algorithm in which the tags frequently used by intimate friends are collected with high probability. It implies that the social tags of a user have weights, hence they can be directly incorporated into our embedding model with the original tags. As depicted in Fig. 2, in the middle layer of our framework, the embbeding vectors of expanded social tags are also be summed with that of original tags to constitute the whole representation vector of a user.

In fact, social tags can be regarded as *collective* tags because they are frequently used by most friends, implying that social tags are more formal than a user's original tags [32]. Therefore, besides profiling users sufficiently, importing social tags can avoid the mismatch of original tags caused by different tagging customs among individuals. Furthermore, our recommendation model can further utilize the social relationships to achieve better performance since the social relationships can also be encoded in the embeddings of social tags, which will be justified in the following experiments.

IV. Evaluation

In this section, in order to justify our model's performance, we implement our model in two real recommendation tasks, i.e., Weibo followship recommendation and Douban movie recommendation.

A. Experimental Settings

1) Datasets: We first introduce how to obtain our experimental datasets[4].

Weibo: Weibo is the largest Chinese Twitter which has attracted more than 0.65 billion accounts and 100 million active users. We first selected 8,000 users as the seeds, and then crawled their followers (about 2.72 million users). There are more than 13.8 million followships discovered. We also crawled original tags of each crawled user. For those users without any tags, we used the method in [32] to select the tags from their social circles as their features. Since a Weibo user does not assign weight for his/her original tags, we set w_i in Eq. 6 to $1/n$ where n is the number of original tags.

Douban: For Douban dataset, we first randomly selected 5,000 Douban users who have rated 15 movies at least. Then, we fetched these users' profiles (including tags), as well as their ratings on their watched movies. All the 5000 users have rated about 42,000 movies in total. We also got all these movies' tags which were labeled by Douban users. Note that Douban user tags have weights but movie tags have no weights. Thus we also set movie tag weights to $1/n$. A Douban movie has a rate ranging from 1 (the least like) to 5 (the most like). If a user rates a movie with score 4 or 5, which implies the user likes the movie, we treat such user-movie pair as a positive sample. In contrast, a dislike case is identified if a user rates a movie by 1 or 2, which is regarded as a negative sample. In general, we can not ensure whether a user likes or dislikes a movie if he/she does not rate the movie at all. It is probably because the user is not aware of the movie or has not yet watched the movie. Therefore, we did not take such cases as negative samples.

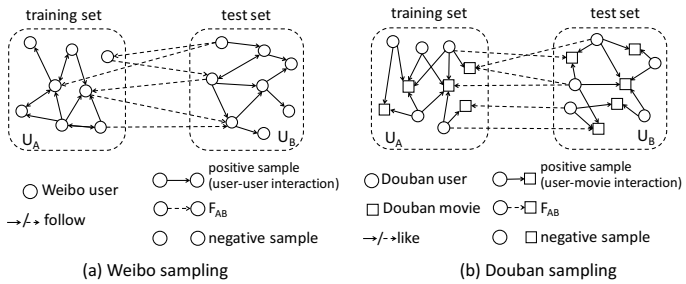


Fig. 3. The separation of training samples and test samples for training the classification model. The user-item interactions between U_A and U_B, i.e., F_{AB}, as represented by dashed arrows, were removed and later added gradually in our experiments in order to simulate the (extreme) scenario of sparse user-item interactions. In Sub-figure (a) of Weibo, the recommended items are also Weibo users representing by circles. In Sub-figure (b), the recommended items are movies representing by rectangles.

2) Sample Collection: To generate training set and test set for the classification model in Phase 2 in Fig. 2, we randomly divided the users and items (an item is also a user in Weibo's task) in each dataset into two disjoint groups of equal size[5] at first. These two groups respectively constitute the training set, namely U_A, and the test set, namely U_B, as illustrated in Fig. 3. Then, given a user u and an item v, the pair $< u, v, 1 >$ is a positive sample, as depicted by a solid arrow in the figure, if u and v are both in either U_A or U_B, and they are linked by a followship or a *like* rating. Similarly, $< u, v, 0 >$ is a negative sample if u and v are both in either U_A or U_B but are not linked (not follow or dislike). According to binary classifier's general setting of the ratio between positive samples and negative samples (1:1), we randomly selected the same number of negative samples as positive samples from the negative sample pool of training/test set. In fact, the positive samples in the test set are just the ground truth in our prediction, i.e., these user-item interactions were supposed to be unknown for any test user.

Furthermore, we removed all user-item interactions between U_A and U_B, which are named as F_{AB} and depicted by dashed arrows in Fig. 3, to simulate an extreme cold-start scenario for each user in U_B. It implies that, we can not learn about any preference of a given test user when we try to recommend some items (followees or movies) to him/her because he/she has no pre-known user-item interactions. Next, we recovered the interactions in F_{AB} gradually and used them to constitute the new positive samples of training set. The recovered interactions are denoted by RF_{AB} and $RF_{AB} \subseteq F_{AB}$. According to our model's principle, as RF_{AB} becomes larger, more

[4]The datasets can be obtained from our lab website *http://gdm.fudan.edu.cn/GDMWiki/Wiki.jsp?page=Network%20DataSet*

[5]We also tested other training-test set splits and got consistent results.

and more test users along with their tags are learned by the model because the relevant representations are fed into the model when training the model. The percentage ratio of recovered user-item interactions is denoted by r_r=$\frac{|RF_{AB}|}{|F_{AB|}}$. In our experiments, we evaluated the performance of our model and the competitors under different r_rs.

3) Baselines: We first introduce some state-of-the-art recommendation algorithms as our model's competitors. For most recommender systems, discovering the similarity between the objective users and potential items is the core principle. A well-formed representation of user/item is the prerequisite to achieve precise recommendation. Hence, we first introduce some previous methods as baselines which represent a user by a vector composed in different fashions.

BOW: The first one is Bag-of-Words (BOW for short) which is a popular model in NLP and can be recognized as a content-based recommendation method. Each user/item is specified directly by a bag of tags since tags are good indicators of user's preferences and item's characteristics. In formal, a user/item is represented by a vector which is the sum of the one-hot vectors of all tags of the user/item. As a result, an entry of representation vector is 1 if the user/item has the corresponding tag, and is 0 otherwise. Obviously, BOW can only find the lexical similarity between the tag sets of users/items. Then, we judge whether a user would follow/like another user/item based on the cosine distance between their representation vectors.

FEBD: We denote the second baseline by FEBD, in which a user is represented by the sum of embedding vectors of his/her preferred items, and the embedding vectors are also learned through Skip-gram model [22]. Similarly, an item is represented by the sum of embedding vectors of the users who like it. According to Word2Vec's principle, user/item's embedding vector are learned based on the cooccurrence relationships of two users or items, i.e, two users like the same item or two items are liked by the same user. As BOW, cosine distance between user/item's representation vectors indicates whether the user would follow/like the user/item or not.

FM: We also selected Factorization Machines (FM for short) [23] as the last competitor since that it has been proved superior to some state-of-the-art CF-based models, such as Biased Matrix Factorization (MF for short) and SVD++. We do not select those MF-based methods because they are generally used to predict user ratings rather than top-n recommendation.

WEBD: Besides user/item tag lists, some other corpora can be used to discover tag co-occurrence relationships which is the prerequisite of learning tag embeddings, i.e., $C(a)$ in Eq. 5. For example, tag cooccurrence relationships can also be found through mapping tags into keywords in the texts of Wikipedia documents. However, Wikipedia corpus usually covers much broader topics rather than the domain of Weibo/Douban. Therefore, we believe that the tag co-occurrence relationships found from Wikipedia corpus should not contribute to downstream recommendation as much as those inherent in Weibo/Douban domain. To justify it, we propose another baseline, namely WEBD, which has the same framework as Fig. 2. But we feed the tag embedding model in Phase 1 with keyword lists of Wikipedia documents instead of user/item tag lists. In Phase 2, we only use original tag embeddings to constitute user/item representations.

EBDT/EBDTE: At last, we use EBDT and EBDTE to denote our recommendation model fed only with original tags and the one fed with both original tags and expanded social tags, respectively. For Douban movie recommendation, we took the tags of the movies liked by a user as the user's social tags since Douban does not have sufficient social circles as Weibo. Similarly, a movie's social tags are filtered from the tags of the users who like the movie.

Obviously, FEBD and FM perform well only when sufficient user-item interactions have been obtained, indicating that they are not fit for the scenario of sparse user-item interactions. Since a Weibo user has 10 original tags at most according to Weibo's setting, we only fed top 10 social tags of most weights in EBDTE. Equally, we only selected top 10 user/movie's original tags with most frequencies in Douban movie recommendation.

4) Performance Metrics: We use some popular metrics for the evaluation of recommender systems, i.e., Prec. (Precision), AP (Average Precision), nDCG (Normailzed Discounted Cumulative Gain). In the following result figures and tables, we report performance scores of top-n recommendations averaged on all users in our test set. Moreover, we also use ROC (Receiver Operating characteristic Curve) and AUC (Area Under the Curve) [16]. The larger AUC a binary classifier gets, the better performance the model has for discriminating both positive samples and negative samples.

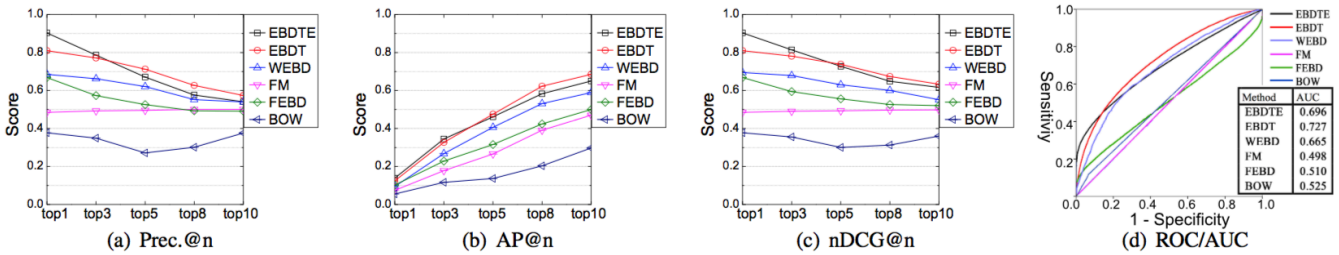
B. Results

1) Study of Embedding Dimension: At first, we study the influence of the dimension of embedding vectors, i.e., k, to classification/recommendation performance. Table I displays EBDT's performance of top-10 Weibo followship recommendation when k varies from 20 to 80 as well as RF_{AB}=$\emptyset$. In addition, the table also lists the time cost of one epoch in training for each case. We pay more attention to the extreme case of RF_{AB}=$\emptyset$ since we propose our recommendation solution addressing the scenario of sparse user-item interactions. As displayed in the following evaluation results, EBDT is superior to the competitors particularly in this extreme sparse scenario. In the table, we only list the results of top-10 recommendation due to space limitation, and other top-n recommendations do not bias the conclusion. From the table we find that the EBDT models of different k exhibit nearly the same performance. Hence we selected k=20 in the following experiments since it takes the least time cost. Moreover, we set M in Eq. 5 to 5 which is the same as generic Skip-gram model [19].

TABLE I
EBDT'S PERFORMANCE AND TIME COST OF TOP-10 WEIBO FOLLOWSHIP RECOMMENDATION ON DIFFERENT DIMENSIONS OF EMBEDDING VECTORS. ALL SCORES WERE EVALUATED IN THE SCENARIO OF $RF_{AB} = \emptyset$, I.E., $r_r = 0\%$.

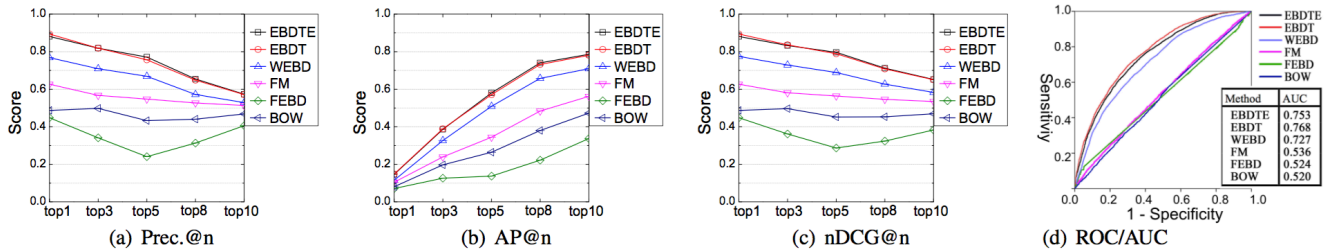
embedding dim.	Prec.@10	AP@10	nDCG@10	time cost
k=20	0.5738	0.6730	0.6261	149.9s
k=40	0.5718	0.6681	0.6231	151.3s
k=60	0.5726	0.6705	0.6247	151.7s
k=80	0.5747	0.6728	0.6260	152.1s

2) Performance Comparisons: As we claimed before, tags characterize users very well, hence we can infer user preferences by tags in stead of user-item interactions when very rare interactions can be obtained, i.e., r_r is very small. Therefore, we first verify the performance of our tag-based models (EBDT/EBDTE) in the scenario of sparse user-item interactions. Specifically, we concern all models' performance in the scenario of r_r=0.1%. Fig. 4 and Fig. 5 display top-1/3/5/8/10



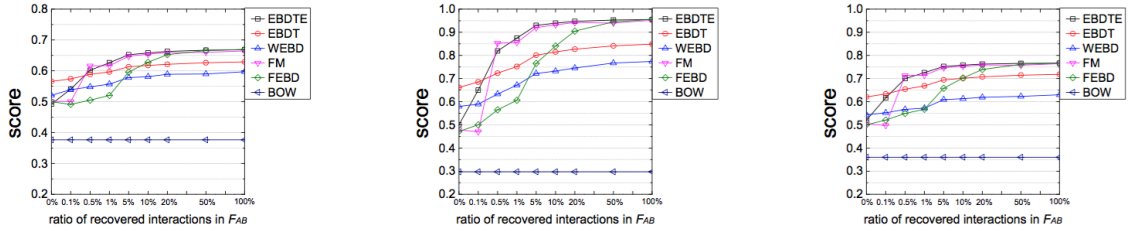
(a) Prec.@n (b) AP@n (c) nDCG@n (d) ROC/AUC

Fig. 4. Performance comparisons of top-n Weibo followship recommendation when recovering 0.1% of the followships in F_{AB}. In Sub-figure (d) of ROC and AUC (Area Under the Curve), sensitivity and specificity are the recall of positive samples and negative samples, respectively. All results show that EBDT and EBDTE outperform all competitors in this scenario of very sparse existing followships.



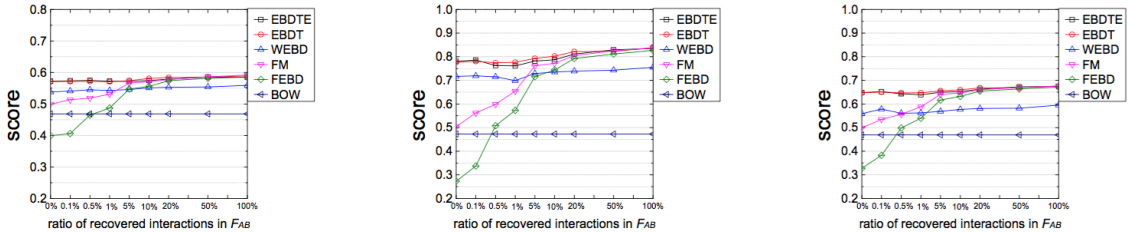
(a) Prec.@n (b) AP@n (c) nDCG@n (d) ROC/AUC

Fig. 5. Performance comparisons of top-n Douban movie recommendation when recovering 0.1% of user-movie relationships in F_{AB}. All results show that EBDT and EBDTE outperform all competitors in this scenario of very sparse existing user-movie relationships.



(a) Prec.@10 (b) AP@10 (c) nDCG@10

Fig. 6. Performance comparisons of top-10 Weibo followship recommendation. The y-axis represents the ratio of recovered followships in F_{AB}, i.e., r_r. The results show that EBDTE and EBDT are superior to their competitors particularly when rare existing followships have been obtained ($r_r < 0.5\%$). EBDTE and FM exhibit nearly the same performance when $r_r \geq 0.5\%$.



(a) Prec.@10 (b) AP@10 (c) nDCG@10

Fig. 7. Performance comparisons of top-10 Douban movie recommendation. The y-axis represents the ratio of recovered user-movie relationships in F_{AB}, i.e., r_r. The results show that EBDTE and EBDT exhibit nearly the same performance in each scenario, and are both superior to the other competitors when less user-movie interactions have been obtained ($r_r < 5\%$).

recommendation performance of Weibo followship recommendation and Douban movie recommendation, respectively. From the figures we find that, for the two recommendation tasks, EBDT and EBDTE outperform all baselines in terms of both top-n recommendation metrics (Prec., AP and nDCG) and binary classification metrics (ROC and AUC). These results verify that the rationale of utilizing tags instead of user-item interactions to profile users/items for data sparsity scenario.

Although BOW also uses tags, it is defeated by EBDT/EBDTE since it only catches the explicit relationships between the two tag lists, i.e., the lexical overlap of two tag sets. This result verifies that tag embedding model discovers the latent correlations between tags very well and hence profiles users/items precisely. BOW keeps the same performance when r_r increases because the tag set of any user/item does not change as r_r changes. Thus, such kind of similarity between a user and an item does not change either.

Furthermore, we explored how user-item interactions help the models gain better performance. For this goal, we evaluated all models' performance when the interactions in F_{AB} have been recovered gradually. Due to space limitation, we only report all models' performance scores on top-10 recommendation as r_r increases from 0% to 100% in Fig. 6 and Fig. 7. The results of other top-n recommendations we have evaluated confirm the same conclusions. From the figures, we find that all models except for BOW increase their performance scores as r_r rises up. Specifically, in Weibo followship recommendation, EBDTE and EBDT exhibit their superiority when $r_r < 0.5\%$. From the scenario of r_r=0.5%, EBDTE and FM have nearly the same performance and outperform their rivals. In Douban movie recommendation, EBDTE and EBDT keep their superior performance steadily in all cases. FM and

FEBD catch up only when $r_r > 20\%$.

3) Discussions: Based above performance results, we give some further discussions.

1. EBDT and EBDTE are very superior to other competitors when less user-item interactions can be obtained ($r_r < 0.5\%$ in Weibo followship recommendation and $r_r < 5\%$ in Douban movie recommendation), verifying their robust performance in such scenario of sparse user-item interactions. It is because our tag embedding models use tags instead of user-item interactions to characterize a user's preferences, thus EBDT and EBDTE do not suffer from sparse interactions. Furthermore, both two models increase their performance as r_r becomes bigger. It is because that the more user-item interactions in F_{AB} are involved when training the classification model, the better the model can learn user/item representations. Comparatively, BOW keeps its performance under all r_rs because it does not use any obtained user-item interactions.

2. In Weibo followship recommendation, EBDTE outperforms EBDT when $r_r \geq 0.5\%$. Such results justify that, the tags distilled from social context specify Weibo users more sufficiently than original tags. Moreover, EBDTE captures the latent social relationships (including predicted followships) between Weibo users. However, in Douban movie recommendation, EBDTE performs nearly the same as EBDT in all scenarios of different r_rs. It is because the expanded tags in this task are not the real social tags of Douban users according to our experiment setting.

3. EBDT's superiority over WEBD proves that, the cooccurrence in user/item tag lists of a given domain are more suitable than the cooccurrence in free texts (Wikipedia documents) to learn better tag embeddings in terms of the recommendation in that domain.

4. Although FEBD and FM have been proved being excellent recommendation model in previous literatures [22], [23], they perform so weakly in the scenario of sparse user-item interactions. It is because they predict the relationship between a target user and a candidate item through the learning based on the existing user-item interactions. Specially as shown in Fig. 4(d) and 5(d), they perform very close to a random guess model (AUC$\approx$0.5) when r_r=0.1%. But FM enhances its performance quickly as more user-item interactions in F_{AB} are recovered, implying that FM can utilize the interactions very well. Nevertheless, in both recommendation tasks, FM can not outperform EBDTE even when all F_{AB} are recovered (r_r=100%), indicating that EBDTE can also exploit obtained interactions excellently as FM.

V. Conclusion and Future Work

Addressing the data sparsity of user-item interactions in recommender systems, we propose a unified recommendation framework based on neural networks. To achieve better recommendation performance, our model first learns tag embeddings based on the Skip-gram model, and then imports social tags for more complete user profiling. The experimental results evaluated on two the real recommendation tasks, i.e., Weibo followship recommendation and Douban movie recommendation, exhibit our model's superiority over the state-of-the-art competitors.

In future, in order to profile users/items more sufficiently, we will continue to explore new methods of tag expansion other than social context, such as utilizing semantic relationships rather than cooccurrence relationships between tags which are extracted from knowledge bases.

References

[1] S. Aciar, D. Zhang, S. Simoff, and J. Debenham. Recommender system based on consumer product reviews. In *Proc. of WI*, 2006.

[2] L. A. Adamic and E. Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211-230, 2003.

[3] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski. Random walks on context spaces: Towards an explanation of the mysteries of semantic word embeddings. *arXiv:1502.03520[cs.LG]*, Mar. 2015.

[4] O. Barkan and N. Koenigstein. Item2vec: Neural item embedding for collaborative filtering. *arXiv:1603.04259v[cs.LG]*, Mar. 2016.

[5] I. Cantador, A. Bellogin, and D. Vallet. Content-based recommendation in social tagging systems. In *Proc. of Recommender System*, 2010.

[6] J. Chen, W. Geyer, C. Dugan, M. Muller, and I. Guy. Make new friends but keep the old, recommending people on social networking sites. In *Proc. of CHI*, 2009.

[7] P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In *Proc. of RecSys.*, 2016.

[8] A. V. den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Proc. of NIPS*, 2013.

[9] X. Dong, L. Yu, ZhonghuoWu, Y. Sun, L. Yuan, and F. Zhang. A hybrid collaborative filtering model with deep structure for recommender systems. In *Proc. of AAAI*, 2017.

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

[11] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367 - 378, 2002.

[12] T. HoFann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 2:89 - 115, 2004.

[13] T. Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proc. of SIGIR*, 2003.

[14] Y. Jia, X. Song, J. Zhou, L. Liu, L. Nie, and D. S. Rosenblum. Fusing social networks with deep learning for volunteerism tendency prediction. In *Proc. of AAAI*, 2016.

[15] Y. Liu, Z. Liu, T.-S. Chua, and M. Sun. Topical word embeddings. In *Proc. of AAAI*, 2015.

[16] G. Mark. Receiver operating characteristic (roc) plots: Fundamental evaluation tool in clinical medicine. *Clin Chem*, 30(4):561 - 567, 1993.

[17] M. McPherson, L. Smith-Lovin, and J. Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27:415 - 445, 2001.

[18] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *arXiv:1301.3781*, 2013.

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

[20] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proc. of Workshop of artificial intelligence and statistics*, 2005.

[21] C. Musto, G. Semeraro, M. de Gemmis, and P. Lops. Word embedding techniques for content-based recommender systems: An empirical evaluation. In *Proc. of RecSys Posters*, 2015.

[22] M. G. Ozsoy. From word embeddings to item recommendation. *arXiv:1601.01356v2[cs.LG]*, Mar. 2016.

[23] S. Rendle. Factorization machines. In *Proc. of ICDM*, 2010.

[24] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proc. of ICML*, 2007.

[25] D. Scherer, A. C. Mller, and S. Behnke. Evaluation of pooling operations in convolutional architectures for object recognition. In *Proc. of ICANN*, 2010.

[26] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie. Autorec: Autoencoders meet collaborative filtering. In *Proc. of WWW*, 2015.

[27] S. Sen, J. Vig, and J. Riedl. Tagommenders: Connecting users to items through tags. In *Proc. of WWW*, 2009.

[28] D. Shin, S. Cetintas, and K.-C. Lee. Recommending tumblr blogs to follow with inductive matrix completion. In *Proc. of RecSys Posters*, 2014.

[29] P. Vincent, H. Larochelle, Y. B. I. Lajoie, and P.-A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *JMLR*, 11:3371 - 3408, 2010.

[30] H. Wang, N. Wang, and D.-Y. Yeung. Collaborative deep learning for recommender systems. In *Proc. of KDD*, 2015.

[31] D. Yang, Y. Xiao, Y. Song, and W. Wang. Semantic-based recommendation across heterogeneous domains. In *Proc. of ICDM*, 2015.

[32] D. Yang, Y. Xiao, H. Tong, J. Zhang, and W. Wang. An integrated tag recommendation algorithm towards weibo user profiling. In *Proc. of DASFAA*, 2015.

[33] Y. Zhang and M. Pennacchiotti. Predicting purchase behaviors from social media. In *Proc. of WWW*, 2013.