Distributed Representations for Content-based and Personalized Tag Recommendation

***Abstract*—We consider the problem of learning distributed representations for documents from their content and associated tags, and of distributed representations of users from documents and tags provided by users. The documents, words, and tags are represented as low-dimensional vectors and are jointly learned with a multi-layered neural language model. We propose a two stage method where in the first stage which consists of two layers, we exploit the corpus wide topic-level information contained in tags to model one layer of neural language model and use document level words sequence information to model other layer of the proposed architecture. In the second stage, we use thus obtained document and tags representations to learn user representations. We utilize these jointly trained vector representations for personalized tag recommendation tasks. Our experiments on two widely used bookmarking datasets show a significant improvements for quality of recommendations. These continuous vector representations has the added advantages of conceptually meaningful which we show by our qualitative analysis on tag suggestion tasks.**

I. Introduction

With the advent of Web 2.0 and explosive growth of user generated content, tagging –a social bookmark indexing activity– has become a preferred medium for conceptually organizing and summarizing information. In most cases, tags reveal thematic information contained in documents at varying granularity and emphasize important keywords (or key-phrases) that may or may not be present in the document. In addition to tags, natural language content of the documents (i.e. word sequences within) present a rich source of semantic information that can be leveraged together with thematic information from tags for representing documents in a meaningful and useful way. Moreover, tags are inherently personalized in nature as they represent preferences and information needs of users who generated these tags. Together with document representations and users' historical tagging activity, users' preferences can also be represented in the same space as document representations that can provide personalized tag recommendations.

In this paper, we present algorithms that can simultaneously model documents from their natural language content as well as from tags, and *represent* these documents along with tags and words sequences in a common low-dimensional vector space. We also present an efficient way to utilize these document and tag representations to derive individual user's vector representations that also sits in the same low-dimensional space as that of documents. As a result, these user representations can be easily combined with document representations to construct personalized features for tag recommendation task. Our algorithms are novel extensions to statistical neural language models [1], [2] for learning these continuous vector representations. Continuous vector (or distributed) representations learned using statistical neural language models have been shown to be successful on various natural language processing related tasks [3]. This is primarily because distributed word representations, such as [4], [2] carry semantic information where words that are contextually similar are "closer" compared to dissimilar ones. For example, distributed representation of word "strong" is closer (in euclidean space) to "powerful" than word "paris". These distributed representations derive their semantic capability from neural language modeling scheme where, given a sequence of words (or *context*), distributed representations of words in the sequence are concatenated to predict the next word. The context level word prediction tasks tend to predict words that are contextually similar such as "strong" and "powerful" by representing them *close* to each other.

Le et al., [2] utilized these word-level distributed representations to construct document level representations where document representation acts as a contextual unit for word sequences present within. The main motivation behind their modeling scheme is that the document representation can act as a contextual unit across multiple sequences (i.e., contexts) and improve prediction of words in different contexts. Although these document representations carry document level semantic information, they lack in corpus level topic information and, therefore, may overfit to particular sequences present in training documents. One potential solution to this problem is to incorporate meta-level information such as tags associated with the documents. Tags can act as empirical prior which can avoid overfitting to a particular sequence within the document, and help learn smooth document representations across similar documents. Since these tags do not come in isolation and are associated with users who generated them, considering users along with tags can provide further improvement in document and tag representations. Furthermore, corpus wide topic level information present in tags can improve the document representations for higher level tasks such as document classification and information retrieval.

Owing to the above intuition, we propose a multi-layered neural language model. Our modeling scheme consists of two stages. The first stage consists of two layers: In the first layer, document representations are learned in conjunction with word representations. This layer gives us a good representations of words while moderate representations of documents. In the second layer, document repre-

sentations are used in combination with the tags associated with the documents to learn tags representations, and further improve document representations. In our task, documents come along with tags, therefore, the probability distribution of observing a word depends not only on some fixed number of surrounding words, but is also conditioned on the specific document. In addition, the probability distribution of observing a document depends on its associated tags. In the second stage, we use document and tag representations to learn user representations. More specifically, we assume a directed relationship between document, tag, and user. The vector representation of a tag can be obtained by summing the vector representation of the document and the vector representation of the user who provided that tag. This approach is similar to the one presented in [5]. Such relationship between tag and document gives us a user vector representation that provide further improvement in tag recommendation by including the personalization factor.

Vector representations learned by our model are useful for various information retrieval and machine learning tasks. For example, with a simple vector similarity operation in the common representation space, a tag recommender system can retrieve relevant tags for previously unseen documents. Similarly, for a given query keyword, most similar tags or words can be retrieved that expand upon the query keyword and present semantic interpretations of the keyword. For machine learning tasks, these representations can be utilized as features for learning a multi-label classifier which can improve on content based tag recommendation task. In fact, our experiments show that the document representations obtained from tags and word sequences, as opposed to traditional tf-idf based representation, can improve upon tag recommendation task by as much as 40% in terms of accuracy of retrieved top tags.

In summary, our main contributions are two fold:

- We propose a neural language model that takes advantage of corpus level topic information contained in tags, document level context information for words (i.e., word sequences) as well as personalized information contained with in user-tag-document triplets, to learn the low-dimensional vector representations of words, document, tags, and users. In the cases when we do not consider tags –either deliberately or in the absence of tags– document vector representations [2] rely solely on document's content level information, which may overfit to specific word sequences contained within documents. On the other hand, when we use tags in modeling representations, topic level information from tags can act as empirical prior for document representations that can help to learn smooth vector representations. Furthermore, considering users along with tags and documents provide the user representations, useful for personalized recommendations.
- We apply our proposed neural language model to tag recommendation tasks and show that the document representations learned by our model can capture semantic information from tags that improves upon content based tag recommendation task very significantly.

II. Related Work

Considering our main goal is to learn neural language modeling based document representations that can improve upon tag recommendation task in bookmarking systems, we present related work in following two areas of research:(1) vector representation learning for documents and associated entities; and (2) Tag recommendations for documents.

A. Neural Language Models and Distributed Representations

The goal of language modeling is to estimate the likelihood of a specific sequence of words appearing in a corpus. Formally, given a sequence of words $\mathbf{W} = \{w_1, w_2, .., w_N\}$ for $w_i \in \mathbb{V}$ (i.e., some vocabulary), we would like to maximize $\sum_n \log\ \mathbb{P}(w_n|w_1, w_2, .., w_{n-1})$ over the sequence. Earlier research in distributed representation learning [1] has focused on using probabilistic neural networks to build general representations of words that improve upon generalization of the classic n-gram language models. Recently, the scope of distributed representations has been extended beyond its original goal of "representing" words for maximizing observed sequence likelihoods to a number of other higher level entities such as sentences and documents [2], relational entities [6], stream of documents [7], social networks [8], and many more. Here, we review recent work in probabilistic neural language models for learning representations that has directly motivated our work.

1) Vector Representation of Words and Documents: In the framework of neural language models, every word w in a corpus vocabulary is mapped to a unique vector v_w which acts as a feature representation for the word. Based upon these features, a simplified neural network (i.e., without any hidden layer) is trained for the classification task of predicting the identity of a word given its surrounding words. Two popular variations of this classification tasks [4] are: (1) continuous bag-of-words model (CBOW); and (2) Skip-gram.

a) CBOW: : CBOW defines a log-linear classifier to predict current word in a sequence based on its preceding and succeeding words, where their representations are averaged as the input. More precisely, the objective of the CBOW model is to maximize the log-likelihood, $\mathscr{L} = \sum_n \mathbb{P}(w_n|w_{n-c}, .., w_{n+c})$ where $(w_{n-c}, .., w_{n+c})$ is a sequence of length $c-1$ and excludes w_n. $\mathbb{P}(w_n|w_{n-c}, .., w_{n+c})$ is defined as:

$$\mathbb{P}(w_n|w_{n-c}, .., w_{n+c}) = \frac{exp(\hat{\mathbf{v}}^\top \mathbf{v}_{w_n})}{\sum_{\mathbf{v}'} exp(\hat{\mathbf{v}}^\top \mathbf{v}')} \qquad (1)$$

Here, $\mathbf{v}_{w_n}$ is the vector representation of the output word w_n and $\hat{\mathbf{v}}$ is the average representation of input sequence, i.e., $\hat{\mathbf{v}} \propto \sum_{-c<j<c; j\neq n} \mathbf{v}_{w_j}$. [2] later extended CBOW model to include sentence representation where $\mathbf{v}_{w_n}$ also includes representation vector for any super sequence (e.g. sentences or paragraphs) which captures sentence level context information. However, note that CBOW and its paragraph level extension take only partial advantage of the word order, since any ordering of a set of contextual words will result in the same vector representation.

b) Skip-gram Model: Another related variant of neural probabilistic modeling scheme, termed as Skip-gram, predicts the surrounding words within a certain sequence based on the current one. Formally, skip-gram defines the objective function as the exact counterpart to the CBOW, i.e., $\mathscr{L} = \sum_n \mathbb{P}(w_{n-c}, .., w_{n+c}|w_n)$, where independence assumption factorizes the probability function as: $\mathbb{P}(w_{n-c}, .., w_{n+c}|w_n) = \prod_{-c<j<c; j\neq n} p(w_{n+j}|w_n)$, with:

$$\mathbb{P}(w_{t+j}|w_n) = \frac{exp(\mathbf{v}_{w_n}^\top \mathbf{v}_{w_{n+j}})}{\sum_{\mathbf{v}'} exp(\mathbf{v}_{w_n}^\top \mathbf{v}')} \quad (2)$$

Similar to sentence level extension of CBOW, Skip-gram's extension replaces input word w_n with sentence level representation vector and constructs a distributed bag-of-words (DBOW) representation for sentence (or paragraph) [2]. Again, similar to CBOW, Skip-gram and its sentence level extension does not take advantage of ordering of the words to its fullest as each word is weighted equally. As a result, one can expect that the sentence (or document) level representation can benefit from extra topic level of information, e.g., tags in our case.

B. Tag recommendation for documents

Tag recommendation approaches can be classified into two main categories: user-centered approaches (i.e., *personalized*) and document-centered approaches (i.e., *content-based*). User-centered approaches take users' historical tagging behaviors into account while suggesting tags for a given combination of document and user. Tensor factorization schemes, such as Canonical decomposition and Tucker Decomposition with optimization over pair of ranking constraints [9], [10], [10], have been shown to outperform traditional graph (e.g. personalized pagerank [11]) and popularity based [12] personalized tag recommendation schemes. Two popular optimization functions for factorization schemes are ranking statistic AUC (area under the ROC-curve) [10] and Bayesian Preference Ranking (BPR) [13]. Both optimization functions treat three-way relationship between user-item-tag as ranking constraints where observed tuples are ranked higher during training than the unobserved ones. In contrast, we employ an energy-based framework where observed tuples are assigned less energy compared to the unobserved ones.

Content based approaches focus on document's content and find association between themes present in the documents and their tags [14], [15]. In contrast to personalized tag recommendation, content based approaches are useful in *cold-start* settings where either user or content does not have any tagging history (e.g., new users or new documents). Various techniques for content based tag recommendations include popularity based [16], tag and term co-occurrences based [17], and machine learning based. A comprehensive review of these techniques can be found in [18]. Machine learning based methods, such as multi-label classification based [15], Sparse Gaussian Processes [19], and two-way Poisson process based [20] have been shown to outperform other heuristic based methods on both the exhaustiveness and efficiency of content based tag recommendation, especially in the cold-start setting. For a comparison of different approaches for content based tag recommendations, we refer readers to [21].

III. Approach

In this section we present our two stage learning approach for computing distributed representations for documents, tags, and users in a shared, low-dimensional embedding space. Our first learning stage is inspired from recently proposed method [2] for learning vector representations of sentences (or documents) which takes advantage of the words order observed in a given sentence (or document). However, in contrast, we also exploit the topic level information contained in tags of documents for learning their representations as well as using them to improve document representations. Note that we do not use any user specific information in our first learning stage. In our second learning stage, we let users induce a functional relationship between documents and tags representations. We utilize an energy-based framework to learn the parameters of these functional relationships which act as user representations in our learning framework.

A. Stage I: Joint representation of documents and tags

We consider the learning setting where we assume that the training documents are annotated with their corresponding tags, e.g., annotated documents from bookmarking websites such as delicious.com. Formally, we are given a corpus $\mathcal{M}$ of M documents $(d_1, .., d_m, .., d_M)$ where each document d_m is annotated with certain tags, i.e. $(t_{m,1}, .., t_{m,T_m})$. Moreover, each document d_m is a sequence of N_m words, i.e. $(w_{m,1}, .., w_{m,N_m})$. Hereafter, we drop document index m from its words and tags indices as it should be clear from the context. We aim to simultaneously learn a D-dimensional continuous feature representations of documents, words, and tags in a common vector space. We propose a two layered architecture where the document

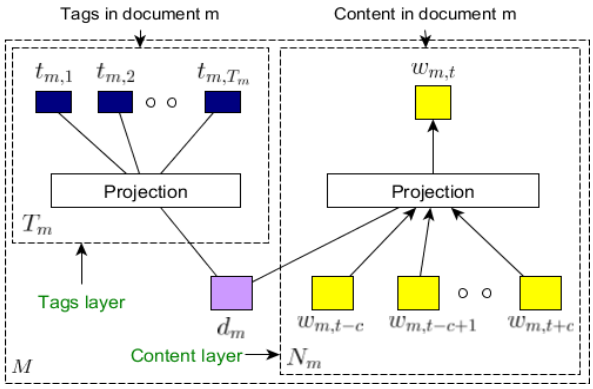

Figure 1: Proposed architecture for tag (blue nodes) specific and content (yellow nodes) specific neural language models with document (purple) nodes as interconnections

vectors predict not only the global context of the words in a document but also the tags associated with the document. The model architecture is shown in Fig. 1, where the first layer models the context of a word in a document given its surrounding words and the global context vector for

the document. The second layer acts as a distributed bag of tags layer where given the global context vector for the document, the presence of its tags is modeled. Formally, the objective function that we try to maximize is:

$$\mathscr{L} = \sum_{d_m \in \mathcal{M}} \Big\{ \sum_{w_n \in d_m} \big\{ \log\ \mathbb{P}(w_n | \mathbf{w}_{\{n-c,\ ..,\ n+c\}}, d_m) \big\} + \alpha \log\ \mathbb{P}(d_m | \mathbf{w}_{\{1,\ ..,\ N_m\}}) + \log\ \mathbb{P}(\mathbf{t}_{\{1,\ ..,\ T_m\}} | d_m) + (1-\alpha) \log\ \mathbb{P}(d_m | \mathbf{t}_{\{1,\ ..,\ T_m\}}) \Big\} \quad (3)$$

Here, c is the length of the context. α corresponds to the weight given to the contribution from content compared to the tags while learning the document embedding jointly [1].

As evident from fig.1, we use CBOW [4] for our content layer however, Skip-Gram architecture can be used as well. Our experiments indicate similar performance with either of the configurations. Note that the tag layer in Fig. 1 resembles distributed Bag of words (DBOW) representation described in [2] with one notable difference that we learn in both directions (i.e., undirected edges between document and tags representations). In practice, it means that for each iteration of stochastic gradient descent, we formulate classification tasks for both the presence of tags given the document as well as the likelihood of observing the document given combinations of tags.

Based upon the architecture illustrated in Figure 1, probability of observing a word depends not only on its surrounding words, but also on the specific document that the word belongs to. More precisely, probability $\mathbb{P}(w_n | \mathbf{w}_{\{n-c,\ ..,\ n+c\}}, d_m)$ is defined as:

$$\mathbb{P}(w_n | \mathbf{w}_{\{n-c,\ ..,\ n+c\}}, d_m) = \frac{\exp(\hat{\mathbf{v}}^\top \mathbf{v}_{w_n})}{\sum_{\mathbf{v}'} \exp(\hat{\mathbf{v}}^\top \mathbf{v}')} \quad (4)$$

where $\hat{\mathbf{v}} \propto (\mathbf{v}_{d_m} + \sum_{-c<j<c; j\neq n} \mathbf{v}_{w_j})$. Conditional probability of observing a tag t given the document m is defined by the soft-max function as given below:

$$p(t_{(m,i)} | d_m) = \frac{\exp(\mathbf{v}_{t_i}^\top \mathbf{v}_{d_m})}{\sum_{t'} \exp(\mathbf{v}_{t'}^\top \mathbf{v}_{d_m})} \quad (5)$$

Similarly, conditional probability of observing a document d_m with the tag t is also defined by the soft-max function as given below

$$p(d_m | t_i) = \frac{\exp(\mathbf{v}_{t_i}^\top \mathbf{v}_{d_m})}{\sum_{d'} \exp(\mathbf{v}_{t_i}^\top \mathbf{v}_{d'})} \quad (6)$$

For defining probability of observing a document given the words contained within, we reuse above equations after replacing the appropriate variables.

1) Learning Vector Representations: We train the network described in Fig. 1 by stochastic gradient descent. The number of parameters [2] in our model are $(M + N + T).D$, where D is the dimensionality of representations and M,N, and T are the number of unique documents, words, and tags respectively. The training algorithm iterates in two stages where we, first, learn the document representations from word context windows, i.e., calculate gradient on content layer in Fig. 1. After we have obtained good initializations for document vectors, we proceed with tag layer in Fig. 1, where we randomly select a fixed set of tags for the document and take a gradient step for both the vector representations of document and tags. This training procedure is akin to pre-training [22] which has proven to be extremely helpful in deep layered neural architecture.

For each individual gradient step, we need following gradients $\nabla\mathbb{P}(w_n | \mathbf{w}_{\{n-c,\ ..,\ n+c\}}, d_m)$, $\nabla\mathbb{P}(d_m | t_i)$, and $\nabla\mathbb{P}(t_i | d_m)$; the computational cost of which is proportional to N, i.e. vocabulary size, M, i.e. number of documents, and T, i.e., size if tags vocabulary, respectively. For large web-scale datasets, the computational cost can quickly become prohibitive. Therefore, an effective alternative is required to reduce the computation for the normalization constant per gradient step. Fortunately, hierarchical soft-max [23] can provide an effective alternate that can reduce the total training computation (per gradient computation) complexity to $\mathcal{O}(c\log(N) + \log(M) + \log(T))$, where c is the length of context window. Instead of evaluating every distinct word, tag or document during each gradient step in order to compute the sums in equations 4, 5, and 6, hierarchical softmax uses three binary trees, one with distinct words as leaves, the second with distinct tags as leaves and the third with distinct documents as leaves. For each leaf node, there is a unique assigned path from the root which is encoded using binary digits. To construct a tree structure, the Huffman encoding is typically used where more frequent words (or tags/documents) in the data set have shorter codes to speed up the training. The internal tree nodes are represented as real-valued vectors, of the same dimensionality as word and document vectors. For example, for computing $\mathbb{P}(d_m | t_{m,i})$ in equation 6, first we convert all documents in the corpus to their Huffman coding representations where we use the number of annotations (i.e., user and tag pair) for each document as its frequency. Hierarchical soft-max expresses the probability of observing the current document given a particular tag as a product of probabilities of the binary decisions specified by the Huffman code of the document as

$$\mathbb{P}(d_m | t_{m,i}) = \prod_l \mathbb{P}(b_l | h_l, t_{m,i}) \quad (7)$$

where b_l is the l^{th} bit in the code with respect to h_l, which is the l^{th} node in the specified tree path of d_m. The probability of each binary decision is defined as follows,

$$\mathbb{P}(b_l = 1 | h_l, t_{m,i}) = \sigma(\mathbf{v}_{t_{m,i}}^\top \mathbf{v}_{h_l}) \quad (8)$$

where $\sigma(.)$ is the sigmoid function and $\mathbf{v}_{h_l}$ is the vector representation of node h_l. Similar analytical expressions follow for calculating $\mathbb{P}(d_m | t_i)$, $\mathbb{P}(w_j | w_i)$, and $\mathbb{P}(d_m | w_i)$. We use total word frequency, tag assignment frequency for constructing corresponding Huffman trees for words and tags respectively.

B. Stage II: Learning User Representations

Our approach for obtaining user representations is inspired by work in energy-based frameworks for learning

[1] we set α to be 0.5 for our experiments

[2] excluding softmax weights

embeddings of entities in low-dimensional spaces [24], [25]. Energy based frameworks for learning entity embeddings have been shown to be successful in representing multi-relational data in Euclidean space such as entity and their relations in Knowledge-Bases [26]. Considering that most recent approaches [25] have focused on increasing the expressivity and the universality of underlying energy based models, majority of these approaches [24], [25] are not suited for user representation learning because of the number of parameters involved (i.e., $O(D * U * (M + T))$, where D is the number of parameters for energy model (typically dimensionality of representations itself) and U, M, and, T are unique users, documents and tags). Fortunately, translation based models [5] provide a suitable framework where the number of parameter only grows as $O(D * (U + M + T))$. In [5], authors consider that a target entity's embedding is obtained by source entity's embedding plus the embedding of the relation between source and target entities. Consider a directed relationship ℓ between two entities h and t [3], then according to [5], $h + \ell$ should be as close as possible to t. Extending this intuition to documents, users and tags, we assume that an embedding of a tag is obtained by summing the embedding of the document and the embedding of the user who provided that tag. The analogy with source-relation-target is as follows: a relationship ℓ when applied to the head word h, converts h to t. Similarly, a user (akin to relation) when reads a document (akin to source word), generates a tag (akin to target word). Now using this analogy we can learn the joint embeddings of users, documents, and tags. However, in order to learn these embeddings, we need to first create supervised data i.e. positive and negative examples. Positive examples are the triplets (u, d, t) where user u provides document d a tag t. Negative examples are the triplets (u, d', t') where user has *not* provided tag t' to document d'. We expand on our sampling procedure for negative examples later in this section. Using this kind of supervised data, we can now formulate the learning problem in the form of the following objective function:

$$\mathcal{L} = \sum_{u,d,t \in S} \sum_{u,d',t' \in S'} [\gamma + d(u + d, t) - d(u + d', t')] \quad (9)$$

where γ is the margin hyperparameter [4], $d(.,.)$ is the distance function between two embeddings, which we take as L-2 norm in our settings. $[x]_+$ denotes the positive part of x, S and S' are the sets containing positive and negative examples respectively.

a) **Selecting training samples:** Whenever a user u creates a tag t for a document d, a triplet (u, d, t) is added to the positive sample set S. For negative samples, for each user u, first we select a tag t' that has never been used by user u from tag's (log) frequency of overall assignment, i.e. $\log(f(t'))$, and then we select a document d' from the set of documents that tag t' is assigned to forming our negative sample (u, d', t'). The reasoning behind taking frequent tags in negative set is to learn user preferences for her own popular tags compared to other popular tags. In our experiments we select 5 such negative examples for each positive example.

Algorithm 1 Personalized Tag Recommendation

Input: Training set $S = \{u, d, t\}$ triplets of users, tags and documents; embeddings of tags and documents
$S_{batch} \leftarrow$ sample(S,b) //Sample a minibatch
$T_{batch} \leftarrow \phi$ //Initialize training set
for Each triplet $\{u, d, t\} \in S_{batch}$ **do** // Generate negative examples
    $T_{batch} \leftarrow \{(u, d', t')\}$
    for $k \in \{1, .., S_{neg}\}$ **do**
        $T_{batch} \leftarrow T_{batch} \cup$ Sample$(S'_{(u,d,t)})$ // Sample with tag assignment frequency weighting
    end for
end for
Update User Embeddings: $\sum_{T_{batch}} [\nabla\gamma + d(u + d, t) - d(u + d', t')]_+$

b) **Learning embeddings::** Although one can optimize (9) for all three embeddings (or distributed representations) i.e., users, documents, and tags as proposed for entity embedding learning in [5], our experiments indicate that the representations learned without incorporating document and tag representations from state I can lead to very poor results. This is mainly due to increased variability induced by users as functional relationships. Entity relationships are governed by relationships that are typically few and provide a reasonable constraint over entities that belong to same functional class. However, user induced functional relationships are rather preferential relationships than factual ones, therefore constraints may not be enough for learning paradigm. Therefore, learning document and tag representation in Stage I serve as a pre-training commonly used in deep neural networks where network weights are first learned in an unsupervised way and later fine-tuned using supervised data. In our approach, we pre-train the tag and document embeddings in a non-personalized way, and then use these embeddings along with the supervised personalized data to obtain user embeddings. The algorithm for learning these embeddings is shown in Algorithm 1.

IV. Experiments

In this section, we describe experimental evaluation of our proposed approach for content based and personalized tag recommendation. Our main hypothesis is that the representations obtained from neural embedding framework are effective as features for machine learning based tag recommendation algorithms.

A. Datasets

We take two publically available datasets [5] for our experiments:

[3] In order to avoid notation cluttering, we will denote the entities and their embeddings by same notations.

[4] We set γ to be 0.01 for our experiments

[5] Datasets used and code are available at https://www.dropbox.com/sh/fti5jaw0hpcovzt/AABMMCByUSOKWM-KcicgSGmra?dl=0

c) CiteULike dataset:: CiteULike is a website for researchers to bookmark scientific references by allowing users to specify their personal tags for their bookmarks. We acquired the tagged data set from CiteULike database dump [6] for over five years from November 15, 2004 to October 30, 2009. We mapped the data set to papers that are indexed in CiteSeerX to extract their titles and abstracts. Each entry of the CiteULike record contains four fields: user name, tag, key (the paper ID in CiteSeer), and creation date. Overall, there are 73,190 entries, with 18,270 distinct papers and 13,607 distinct tags. The average number of tags per document is 3.45, with maximum of 125, minimum of 1, and a standard deviation of 3.31.

d) Delicious dataset:: Del.icio.us is one of the largest web2.0 web sites that provides services for sharing webpages as bookmarks among users. For our experiments, we utilize publically available benchmarked dataset from [16] to construct a subset of *DeliciousT140* dataset [7]. *DeliciousT140* dataset is constructed by taking the 140 most popular tags (*T140*) from the Delicious.com site, and collecting latest posts for each tag in the T140 set, which resulted in 379,931 unique documents. After that, all these documents are crawled in order to get their content. Based upon the language filtering (i.e. removing all non-English posts) resulting data set contains 144,574 documents, on which 67,104 different tags had been observed. We selected 5 days (April 07 - 11, 2008) of bookmarking activity from this dataset as our training data and next 3 days (April 12 - 14, 2008) as our test data. Similar to CiteULike dataset, each entry in our record contains four fields: user name, tag, document id, and creation date. Overall, there are 332,885 entries, with 15,973 distinct webpages and 9,175 distinct tags. The average number of tags per document is 9.39, with maximum of 206, minimum of 2, and a standard deviation of 10.05 tags.

B. Experimental Settings and Evaluation Metrics

We use standard information retrieval based metrics for our tag recommendation task. Specifically, we use Precision@K, Recall@K, F-measure@K for our evaluation criteria. These metrics are defined as:

$$P@K = \sum_{d_m} \frac{\#\text{ Correct tags in top K tags in document } d_m}{K}$$

$$R@K = \sum_{d_m} \frac{\#\text{ Correct tags in top K tags for document } d_m}{\text{Total number of tags}}$$

$$MAP = \sum_{d_m} \frac{\sum_{t=1}^{T_m} P@t \times rel(t)}{\text{Total number of tags for } d_m}$$

where $rel(t)$ is 0 or 1 depending on whether retrieved tag t for document d_m is an actual tag or not. For our experiments, we randomly partition CiteULike into 75%/25% splits as our training and testing sets. We prepare 4 such folds and report the averages. For Delicious dataset, we partition it temporally as mentioned above. For our neural language modeling based algorithms, we set the dimensionality of continuous vector space to be 100 unless specified otherwise.

C. Tasks, Baselines, and Results

1) Content based tag recommendation: For our content based tag recommendation task, our main goal is to jointly learn vectors representations of documents, tags, and words from the content of documents and tags assigned to them. For evaluating the quality of these representations, we undertake content based tag recommendation task [14]. The evaluation task is to rank tags of test documents and measure the quality of top ranked tags against documents' actually assigned tags. The two popular tag ranking techniques that are shown to be effective [21] for quality tag recommendations are namely, collaborative filtering based and multi-label classification based. Collaborative filtering based tag ranking methods, first, select most similar k training documents (i.e. k-NN training points) to a given test document and then rank their tags as the recommended tags [27]. Multi-label classification methods [15], [14] typically learn a one-vs-rest classifier on the training documents and, for a test document, tags are recommended based upon ranking imposed by all the tag (or tag category) specific classifiers. Our evaluations center around the effectiveness of underlying representations for these ranking methods where we compare our tags2vec representation against various state of art representations.

a) Representations Compared: For our baselines representations, we consider three types of representations: (1) vector space based (i.e., tf-idf), (2) topic models based [28] where n-topic LDA model is trained and posterior probability $p(topic|doc)$ is taken as document's representation [8], and (3) neural language modeling based representations, that is doc2vec [2], and tags2vec (this paper).

b) Ranking Algorithms Compared: Selection of our ranking algorithms for tag recommendations is inspired by [14] because of their state of art content based tag recommendation algorithms. In particular, we consider three machine learning based algorithms: (1) collaborative filtering based, i.e. content similarity induced k-NN tag ranking (**Sim**) [14], (2) SVM-Struct (**SVM**) [29]; (3) Sparse-Gaussian Processes (**SGP**) [19].

c) Results: Fig 2 and 3 depict the tag retrieval results (P, R, and F @ K) for CiteULike and Delicious datasets respectively. Specifically, we compare the above mentioned ranking algorithms with their respective representation (as reported in the corresponding literature) against tag2vec representation. As evident, tags2vec clearly outperforms baseline representations on both the dataset for all the three algorithms with F-measure in the margin of 40% to 60% accuracy improvements for CiteULike dataset and 20% to 30% for Delicious dataset. It validates our assumption that tags2vec representation for documents captures semantic information from words

[6] http://www.citeulike.org/faq/data.adp

[7] available at http://www.zubiaga.org/datasets/delicioust140/

[8] here, n is calculated as described in [14] for scalability reasons, i.e., set of most frequent tags for each document

	Sim (LDA)	Sim (tags2vec)	SVM (tf-idf)	SGP (tf-idf)	SGP (tags2vec)	SGP (tags2vec) (w/o content)	SGP (doc2vec)	SGP (doc2vec) (content + tags)
CiteULike	0.2063	0.3165	0.3380	0.3616	**0.5353**	0.4857	0.4001	0.3842
Delicious	0.2207	0.3199	0.3947	0.3643	**0.5035**	0.4712	0.4146	0.4093

Table I: Mean Average Precision (MAP) for various representations based upon SGP ranking algorithm.

Title	**Actual Tags**	**Suggested Tags**
Maude versus Haskell: an Experimental Comparison in Security Protocol Analysis	comparison, protocol, formal, security, haskell	security, haskell, programming, concurrency, functional
Middleware and Application Adaptation Requirements and their Support in Pervasive Computing	middleware, requirements, pervasive, dynamic, adaptation, systems	architecture , pervasive , middleware, ubicomp , grid , mobile
What can you do with a Web in your Pocket?	pagerank, personalization, google, search, metadata, web, www	search , information_retrieval , ir, web , information , query , semantic_web
Building a Large Annotated Corpus of English: The Penn Treebank	nlp, natural_language_processing, ontology, annotation, informationretrieval	nlp , text , information_retrieval, parsing , ontology , information , search
Efficient Stochastic Part-of-Speech Tagging for Hungarian	pos, hmm	model , learning , clustering , markov, modeling , algorithm , machine_learning

Table II: Top actual tags and most similar tags for selected documents. Similarity is cosine similarity in shared continuous vector space

	CiteULike						Delicious					
	P@1	P@10	R@1	R@10	F@1	F@10	P@1	P@10	R@1	R@10	F@1	F@10
doc2vec (content + tags)	0.1823	0.0641	0.1651	0.5060	0.1640	0.1114	0.5362	0.2835	0.0714	0.4823	0.1282	0.3177
doc2vec	0.2155	0.0699	0.1951	0.5060	0.1840	0.1374	0.5597	0.32843	0.0826	0.5108	0.1407	0.3451
tags2vec (w/o content)	0.2892	0.0734	0.2362	0.6123	0.2534	0.1543	0.5775	0.3685	0.0920	0.5782	0.15702	0.4191
tags2vec	**0.3638**	**0.1150**	**0.3050**	**0.8777**	**0.3225**	**0.1985**	**0.5959**	**0.4127**	**0.1084**	**0.6355**	**0.16541**	**0.4340**

Table III: Evaluation results for different representations on *CiteULike* and *Delicious* datasets. Results correspond to SGP ranking algorithm.

sequence as well as thematic information from their associated tags. Documents with similar tags as well as similar words sequences should be semantically close and the closeness is exploited not only by the k-NN based tag recommendation method but also by multi-label classification schemes such as Sparse-GP and SVM-Struct. MAP results for all the representations (i.e. language model and neural language model basaed) are shown in Table I.

We also compare tags2vec model based document representations with other neural language modeling based representations. Specifically, we compare tags2vec based document representations against Distributed Memory Model of Paragraph Vectors (*doc2vec*) [2] and a variant of tags2vec model that does not model the content of documents. With respect to the architecture in Fig. 1, doc2vec refers to the content layer whereas *tags2vec w/o content* refers to tags layer. The motivation behind these two representations as baselines is to validate our assumption that the joint learning of document representation from tags and content combines the semantic information contained in word sequences with the topic level information contained in tags, which should provide better document representations compared to modeling from either source of information alone. We also include doc2vec representation for modified input where we simply concatenate tags as keywords at the end of the document, which we refer as *doc2vec(content+tags)*. Results are reported in Table III for the two datasets. Here due to space restrictions, we only show results with K = 1 and 10. Both *doc2vec* and *tags2vec w/o content* perform inferior to tags2vec representation which is mainly because of lack of smoothing from empirical observation of corpus level tags (for doc2vec) and document level content (for tags2vec w/o content) information respectively. Interestingly, tags2vec w/o content performs close to tags2vec for P@1 however deteriorated significantly at P@10. This is mainly because top tags for documents are typically associated with more documents compared to low ranked tags which are learned easily by the neural models whereas the content oriented lower ranked tags are learned better with complete neural architecture (i.e. tags2vec). Results show that the layered approach perform much better than either simply concatenating tags to the content of documents or removing the content altogether.

Due to the fact that document and tags reside in same feature space, qualitatively speaking, we can retrieve similar tags simply with cosine similarity operations in the underlying continuous feature space. Using the trained tags2vec model, we retrieved the nearest tags given a document embedding vector. Table II shows 5 randomly selected papers from our CiteULike dataset with their corresponding actual and retrieved tags. We can observe that the corresponding tags not only matches the actual tags but also summarizes the document as well. For cases where there is only a limited number (upto 2) of tags (e.g. for last document), suggested tags further expand upon nature of the content of the article.

2) Personalized Tag Recommendation Task: We now include the personalization factor in the tag recommendation and consider the personalized tag recommendation task. As mentioned earlier, in personalized recommendation we learn the embedding of users along with the embedding of tags, documents and words. For this task, we experiment on the same two datasets i.e. delicious and citeULike. For these two datasets, we compare our approach, termed as Embedded Personalized Tag Recommendation (EPTR)

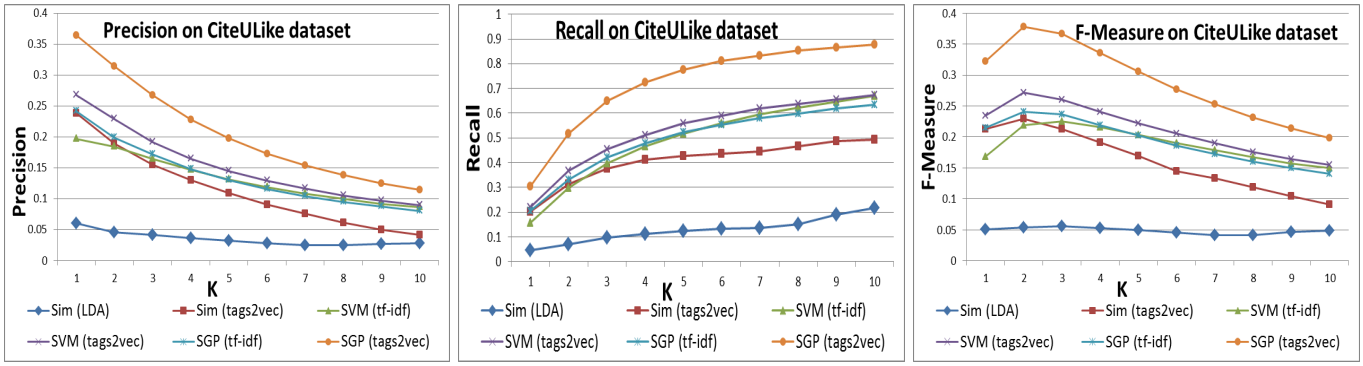

Figure 2: Comparison of baseline ranking algorithms on CiteULike dataset. Here, *Sim* represents content similarity based, *SVM* and *SGP* represents Support Vector Machine and Sparse Gaussian Processes based ranking algorithms. *LDA, tf-idf*, and *tags2vec* represents different representation mentioned in text. In legend, *Sim(LDA)* represents similarity based ranking algorithm with *LDA* based underlying representation.

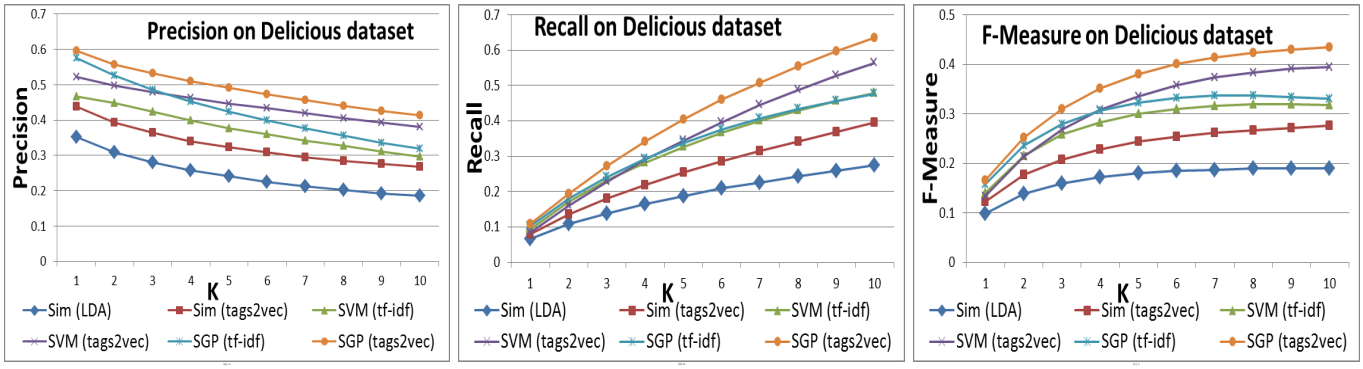

Figure 3: Comparison of baseline ranking algorithms on Delicious dataset. Please see above figure for legend explanations.

with following baselines:

1) TransE [5]: This method was originally proposed to learn the embedding of entities and their relationships. We apply this approach for the tag recommendation task, more specifically to learn the embedding of documents, tags and users. We learn the embedding of users (akin to relations), document (akin to source word), and tag (akin to target word) assuming that there is a directed relationship between document and tag which is encoded by user embedding. Embedding of users, documents and tags are learned simultaneously. It may be worth noting here that in our personalized tag recommendation task (i.e., EPTR) we do not learn the embedding of tags and documents, but only the embedding of users. The embedding of documents and tags are learned from the stage I described in Section III-A.
2) Factorization Machines (FM) [30]: Factorization machines are general predictor like support vector machines. They model all nested variable interactions (comparable to polynomial kernel in SVMs) but uses a factorized parametrization instead of a dense parametrization like in SVMs. Unlike SVMs, the model equation of FMs can be computed in linear time.
3) Pairwise Interaction Tensor Factorization (PITF) [31] [9]: This model is a special case of the Tucker Decomposition (TD) model with linear runtime both for learning and prediction. PITF explicitly models the pairwise interactions between users, items and tags. The model have been shown to outperform TD largely in runtime and even can achieve better prediction quality. It has also won the ECML/PKDD Discovery Challenge 2009 for graph-based tag recommendation.

For both EPTR and TransE, we construct a supervised learning task similar to our content based tag recommendation task. However, instead of document representations as features, we use representations based features for posts (i.e., combination of user and document) by simply adding document and user representations and use normalized vector as a feature. For our personalized recommendation task, although we can train any content based classifier described in previous section, we train SVM based one vs rest multi-lable classifier with linear kernel and margin parameter $C = 1$. The selection of SVM based classifier is mainly due to balance between accuracy and computational complexity.

[9]For both PITF and FM, we use code provided at: http://www.informatik.uni-konstanz.de/rendle/software/tag-recommender/ We set the parameters of these methods comparable to EPTR, i.e. # factors = 100, and # Negative samples = 5

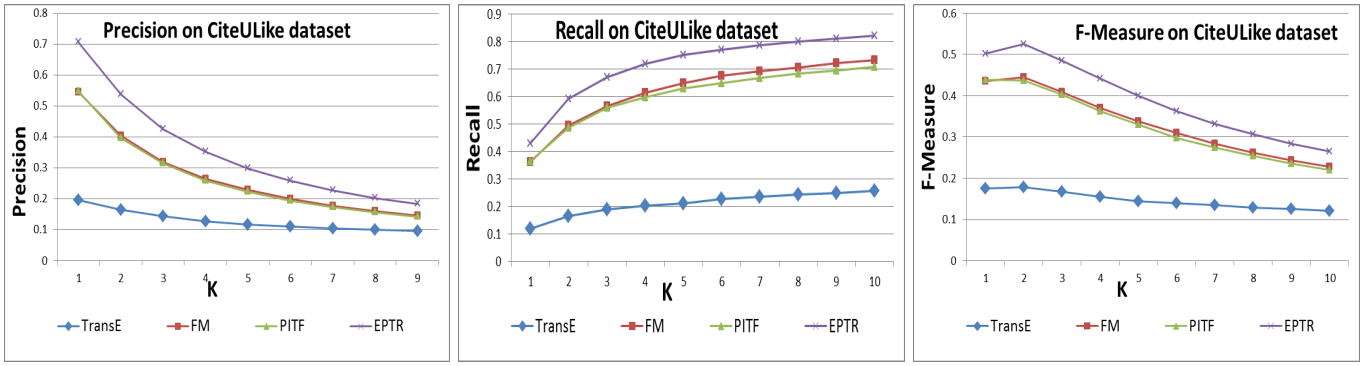

Figure 4: Comparison of baseline ranking algorithms on CiteULike dataset for personalized tag recommendation task, as we vary K for Recall@K, Precision@K and F-measure@K.

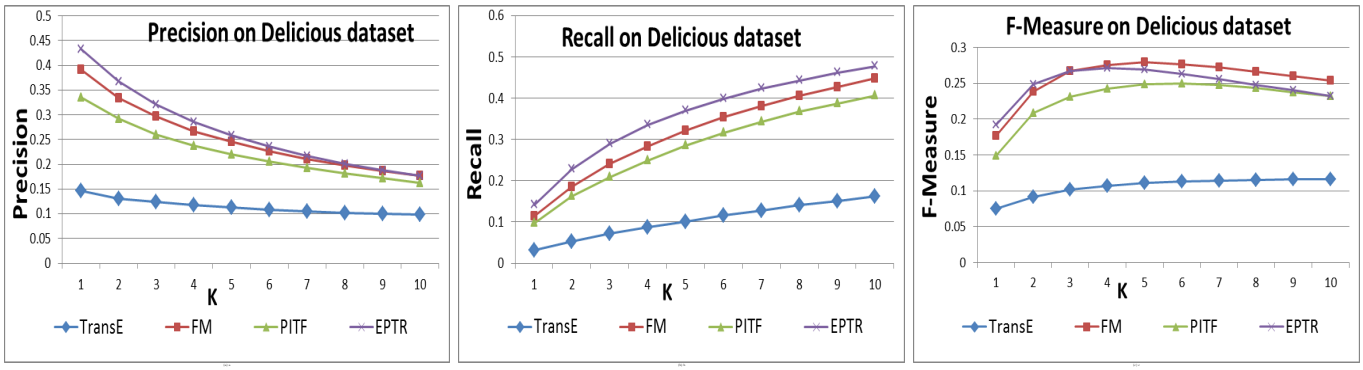

Figure 5: Comparison of baseline ranking algorithms on Delicious dataset for personalized tag recommendation task, as we vary K for Recall@K, Precision@K and F-measure@K.

Results on both datasets are shown in Table IV. In the table, we show the mean average precision (MAP) for all baselines and the propose method. As we see the proposed method outperforms all baselines by a significant margin. For both datasets, our method performs at least 25% better than the best performing baseline.

	TransE	FM	PITF	EPTR
CiteULike	0.28300	0.54367	0.52533	**0.68488**
Delicious	0.19791	0.31290	0.28038	**0.36566**

Table IV: Baseline mean average precision (MAP) comparison for personalized tag recommendation task

We further show the results of the proposed method compared to the baselines as we vary k for recall, precision and F-measure. The results are shown in Figures 4 and 5 for citeULike and Delicious datasets respectively. As we see from Figure 4, our method outperforms the baselines for all three metrics for all values of k. While for the Delicious dataset, our method outperforms all baselines for all three metrics for all values of k except in Precision and F-measure of higher values of k than 4. Results for CiteULike dataset for both content based and personalized tag recommendations are predominantly better than Delicious dataset mainly because of the content information presented to stage I of training is much more clean and thematically coherent. The abstract of documents contain thematically coherent content both in nature of topics covered and quality (i.e., word sequence information) whereas content of webpages in Delicious data contains various noises such as parsing errors as well as intermingled web snippets. However, as evidenced in MAP results in table IV, layered neural language approach beats state of art factorization models for overall retrieval results.

Lastly, we would like to comment upon the time complexity and effect of size of dimensionality on training time and performance respectively. The time complexity of our learning stage I is $\mathcal{O}(D * I * (c * \sum_m N_m * \log(N) + M * \log(M) + \sum_m T_m * \log(T)))$, where c is the length of context window and N, M, T are the total number of unique words, documents, and tags in the corpus. D and I are size of dimensionality and number of iterations, respectively. For our learning stage II, we use minibatch stochastic gradient descent which is upper bounded by $O(D*I*(\#triplets))$, triplets are observed user-document-tag pairs. Both the stages are easily parallelizable and we use a 16-core intel processor for our c-extension of python based code. Moreover, results of first stage can be computed once and saved to be used for next stage. We also vary the dimensionality of our vector representations and observe consistent improvements till # dimensions 100 and then a plateau in our MAP statistics. Therefore, we select 100 as standardized number of dimensions across various baselines.

V. Conclusion

We have presented algorithms that take advantage of tags associated with documents to find more meaningful representation of documents and tags, and at the same time, can find users' representations considering a directed relationship between documents and tags. We have presented a two stage approach where in the first we learn words, documents, and tag representations while in the second stage we learn the users representations. Our experiments show that such representations better capture the semantic and personalized information contained within the documents and tags which results in significant improvement (upto 40% over baselines) in content tags recommendation tasks, and for over 25% in personalized tag recommendation task.

References

[1] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, pp. 1137–1155, 2003.

[2] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, 2014, pp. 1188–1196.

[3] L. Deng and D. Yu, *Deep Learning*. Now Publishers Incorporated, 2014.

[4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

[5] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems*, 2013, pp. 2787–2795.

[6] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in Neural Information Processing Systems*, 2013, pp. 926–934.

[7] N. Djuric, H. Wu, V. Radosavljevic, M. Grbovic, and N. Bhamidipati, "Hierarchical neural language models for joint representation of streaming documents and their content," 2015.

[8] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[9] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "Tag recommendations based on tensor dimensionality reduction," in *Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 2008, pp. 43–50.

[10] S. Rendle, L. Balby Marinho, A. Nanopoulos, and L. Schmidt-Thieme, "Learning optimal ranking with tensor factorization for tag recommendation," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 727–736.

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

[12] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme, "Tag recommendations in folksonomies," in *Knowledge Discovery in Databases: PKDD 2007*. Springer, 2007, pp. 506–514.

[13] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, pp. 452–461.

[14] Y. Song, L. Zhang, and C. L. Giles, "Automatic tag recommendation algorithms for social recommender systems," *ACM Trans. Web*, 2011.

[15] P. Heymann, D. Ramage, and H. Garcia-Molina, "Social tag prediction," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2008, pp. 531–538.

[16] A. Zubiaga, A. Perez Garcia-Plaza, V. Fresno, and R. Martinez, "Content-based clustering for tag cloud visualization," in *International Conference on Advances inSocial Network Analysis and Mining, 2009. ASONAM'09.* IEEE, 2009, pp. 316–319.

[17] B. Sigurbjörnsson and R. Van Zwol, "Flickr tag recommendation based on collective knowledge," in *Proceedings of the 17th international conference on World Wide Web*. ACM, 2008, pp. 327–336.

[18] M. Gupta, R. Li, Z. Yin, and J. Han, "Survey on social tagging techniques," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 1, pp. 58–72, 2010.

[19] Y. Song, L. Zhang, and C. L. Giles, "A sparse gaussian processes classification framework for fast tag suggestions," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 93–102.

[20] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles, "Real-time automatic tag recommendation," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '08. ACM, 2008, pp. 515–522.

[21] J. Illig, A. Hotho, R. Jäschke, and G. Stumme, "A comparison of content-based tag recommendations in folksonomy systems," in *Knowledge Processing and Data Analysis*. Springer, 2011, pp. 136–149.

[22] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *The Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[23] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proceedings of the international workshop on artificial intelligence and statistics*. Citeseer, 2005, pp. 246–252.

[24] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *Conference on Artificial Intelligence*, no. EPFL-CONF-192344, 2011.

[25] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data," *Machine Learning*, vol. 94, no. 2, pp. 233–259, 2014.

[26] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction," *arXiv preprint arXiv:1503.00759*, 2015.

[27] Y.-T. Lu, S.-I. Yu, T.-C. Chang, and J. Y.-j. Hsu, "A content-based method to enhance tag recommendation," in *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, ser. IJCAI'09, 2009.

[28] R. Krestel, P. Fankhauser, and W. Nejdl, "Latent dirichlet allocation for tag recommendation," in *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 61–68.

[29] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output spaces," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 104.

[30] S. Rendle, "Factorization machines," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 2010, pp. 995–1000.

[31] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 81–90.