Neurocomputing 314 (2018) 479–489

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Guiding supervised topic modeling for content based tag recommendation

Yong Wu[a], Shengqu Xi[a], Yuan Yao[a,*], Feng Xu[a], Hanghang Tong[b], Jian Lu[a]

[a] National Key Laboratory for Novel Software Technology, Nanjing University, China
[b] Arizona State University, USA

ARTICLE INFO

Article history:
Received 13 October 2017
Revised 2 May 2018
Accepted 3 July 2018
Available online 17 July 2018

Communicated by Tao Li

Keywords:
Tag recommendation
Similar words
Relevant words
Supervised topic modeling
Generative model

ABSTRACT

Automatically recommending suitable tags for online content is a necessary task for better information organization and retrieval. In this article, we propose a generative model SimWord for the tag recommendation problem on textual content. The key observation of our model is that the tags and their relevant/similar words may have appeared in the corresponding content. In particular, we first empirically verify this observation in real data sets, and then design a supervised topic model which is guided by the above observation for tag recommendation. Experimental evaluations demonstrate that the proposed method outperforms several existing methods in terms of recommendation accuracy.

1. Introduction

Tagging is a widespread mechanism on the Web. On one hand, tags usually indicate the keywords to describe and summarize the online content, which could benefit the organization and retrieval of online content. On the other hand, over 50% online content lacks tag information or even does not have tags at all [18]; moreover, it is often painstaking for users (even the content creators) to manually tag the online content, especially under many situations where the users are not certain about what the appropriate tags are. Therefore, automatically recommending suitable tags for online content becomes a necessary task.

Roughly speaking, existing tag recommendation methods can be divided into the personalized collaborative filtering methods and the object-centered content based methods [2]. In collaborative filtering methods, the key idea is to employ the users' historical behavior, and recommend subjective and personalized tags; in contrast, the key idea of content based methods is to extract the keywords from the content, and recommend objective and general tags (see Section 6 for a review). In this work, we focus on the content based tag recommendation methods for textual content. That is, our aim is to provide objective and general tags for online textual content.

The key idea of existing content based methods is to recommend candidate tags based on the content features, with the implicit assumption that the most relevant information (i.e., tags) are already included in the content. In this work, we explicitly model such assumption in the model, with the empirical observation that many tags have appeared in the corresponding content. Moreover, we observe that the content usually contains not only the tags but also their relevant/similar words. We name such phenomenon as *tag-content relevance*, which can be leveraged to improve the tag recommendation accuracy. Fig. 1 presents an illustrative example of tag-content relevance from Stack Overflow. As we can see from the figure, all the tags (e.g., 'polymorphism' and 'generics') have directly appeared in the post content; additionally, some relevant words (e.g.,'subclass' and 'polymorphic' are relevant to 'polymorphism') have also appeared in the post content.

Based on the above observation, we propose a generative model for the tag recommendation problem, and use the tags as well as their relevant words as the guiding information to improve the performance of tag recommendation. In particular, we first empirically verify the existence of tag-content relevance phenomenon in two real data sets from two different domains (i.e., a programmer community Stack Overflow and a math community Mathematics Stack Exchange). Next, we propose a supervised topic model SimWord, which generates the content words by either the normal

* Corresponding author.
E-mail addresses: wy@smail.nju.edu.cn (Y. Wu), nju.cellzero@gmail.com (S. Xi), y.yao@nju.edu.cn (Y. Yao), xf@nju.edu.cn (F. Xu), hanghang.tong@asu.edu (H. Tong), lj@nju.edu.cn (J. Lu).

https://doi.org/10.1016/j.neucom.2018.07.011
0925-2312/

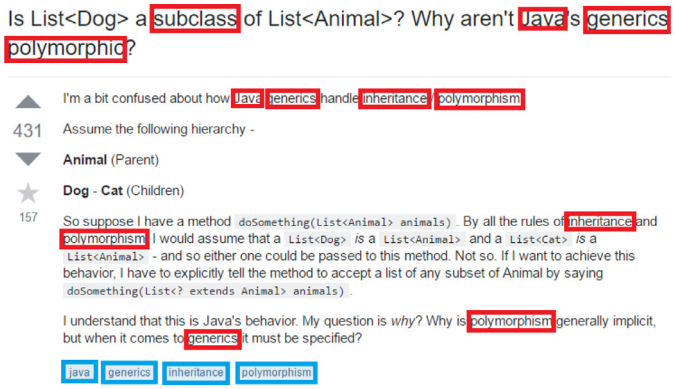


Fig. 1. An illustrative example for tag-content relevance.

tag-word distribution[1] or the relevant-word distribution (serving as guiding information). In other words, while existing topic models generate a word from the normal tag-word distribution, we allow SimWord to generate a word from the tags and their relevant words (with a probability that can be learned for different domains). Finally, we conduct extensive experiments on real data sets to validate the usefulness of employing the tag-content relevance phenomenon, as well as the superior performance of the proposed SimWord method for content based tag recommendation. For example, compared with the existing best competitors, the proposed SimWord can achieve up to 17.1% improvement in terms of recommendation accuracy, while enjoying linear scalability.

The main contributions of this article are summarized as follows:

- We empirically verify the tag-content relevance phenomenon in two real data sets from different domains. We find that tags and their relevant words appear (multiple times) in both the title and the body of the content.
- We propose a generative model SimWord for recommending tags on textual content. The proposed model is built upon the LLDA [25] and further integrates the tag-content relevance phenomenon. We also propose several special cases of SimWord by using the content title only, or by ignoring the relevant words.
- We perform experimental evaluations on four data sets to demonstrate the effectiveness and efficiency of the proposed method.

The rest of the article is organized as follows. Section 2 empirically studies the tag-content relevance phenomenon. Section 3 describes the proposed SimWord model, and Section 4 presents the inference algorithms for the model. Section 5 presents the experimental evaluations. Section 6 reviews related work, and Section 7 summarizes the article with conclusions.

2. Empirical study

In this section, we empirically study the tag-content relevance phenomenon in two real data sets Stack Overflow[2] (SO) and Mathematics Stack Exchange[3] (Math). Both data sets are officially published and publicly available.[4] For each data set, it contains question posts and their corresponding tags. Each question post contains a title and a body. Next, we need some preprocessing on the data sets. For the posts, we remove the stopwords and some low frequency words to reduce noise. We deliberately keep those words that are tags (e.g., 'C' or 'VB'). All the remaining words are then stemmed. For the tags, we remove some low frequency tags as they are seldom used. The statistics of these two data sets can be found in Table 2 (in the experimental section).

A. Tag-content occurrence study. We first study a special case (tag-content occurrence) of the tag-content relevance phenomenon. That is, we study the degree/percentage of tags that have exactly appeared in the corresponding content. If all the tags of a post have appeared in its post content, the degree would be 100%. The results are shown in Fig. 2, where the x-axis indicates the degree of tag-content occurrence, and the y-axis indicates the proportion of posts with the corresponding tag occurrence degree. In the figures, two data sets exhibit different patterns. For example, over 70% posts (79.0%) in SO data have at least one tag appeared in the content, while less than 30% posts (24.8%) in Math data have the same phenomenon. Moreover, there are over 30% posts (33.8%) in the SO data that have all tags appeared in the content. We also study the degree of tag-content occurrence when only the title of the post is used as content (the figures are omitted for brevity). We found that 65.3% posts in SO data and 15.4% posts in Math data have at least one tag appeared in the title.

Next, as mentioned before, the tags may appear multiple times in the content. Therefore, we also study this phenomenon by plotting the proportion of posts against the number of content words that are tags. The results are shown in Fig. 3. A long-tailed distribution can be observed from the figures. For example, in SO data, 47.5%/28.2%/8.9% posts have at least 3/5/10 words that are tags. For some posts, there are even 50 words that are tags.

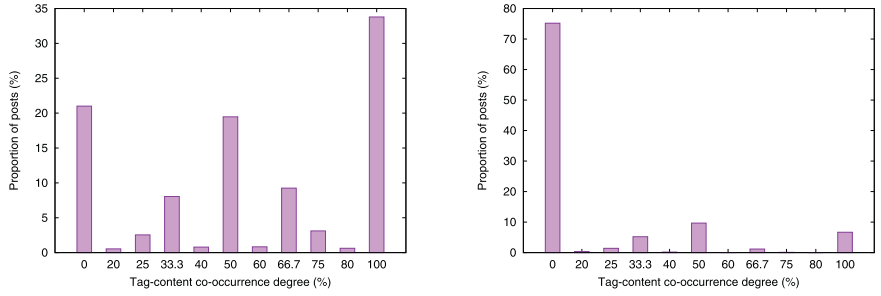
B. Tag-content relevance study. Next, we check the more general tag-content relevance phenomenon. That is, we study the occurrence degree and times of tags as well as their relevant words in the corresponding content. To obtain the relevant words of tags,

[1] The tag-word distribution or the topic-word distribution can be learned by topic models.

[2] http://stackoverflow.com/

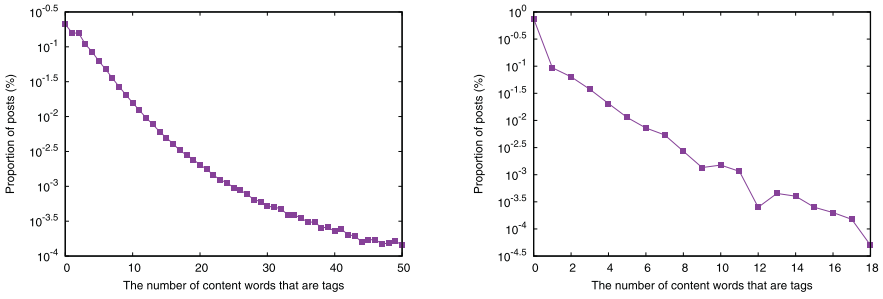
[3] http://math.stackexchange.com/

[4] http://blog.stackoverflow.com/tags/cc-wiki-dump/



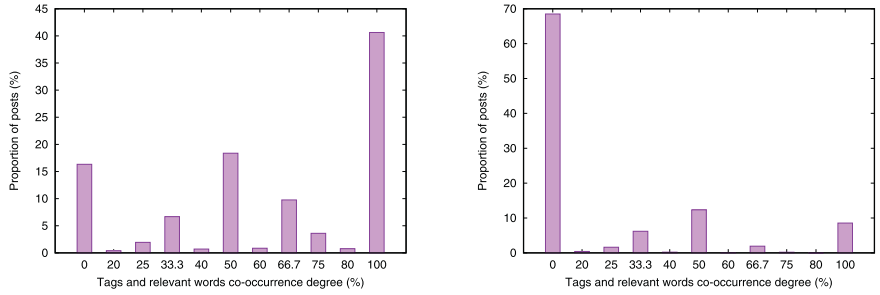
(a) SO data (b) Math data

Fig. 2. The proportion of posts vs. tag occurrence degree.



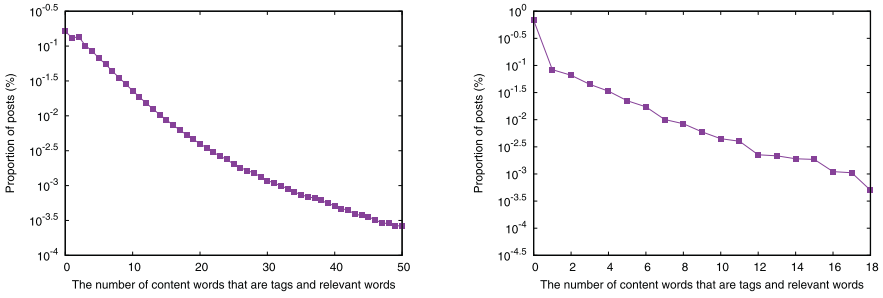
(a) SO data (b) Math data

Fig. 3. The proportion of posts vs. the number of content words that are tags.



(a) SO data (b) Math data

Fig. 4. The proportion of posts vs. tag-content relevance degree.



(a) SO data (b) Math data

Fig. 5. The proportion of posts vs. the number of content words that are relevant words to tags.

we adopt the word2vec model [19]. Based on the output word vectors from word2vec, we choose the top-10 similar words (defined under cosine similarity) as the relevant words for each tag.

Similar to Figs. 2 and 3, the results of tag-content relevance phenomenon are shown in Figs. 4 and 5, respectively. In general, we can observe similar distributions when we take relevant words into consideration, and the occurrence degree and times are higher than those in Figs. 2 and 3. Take the SO data as an example. Less than 35% posts have all their tags in the content, while more than 40% posts have all their tags or relevant words in the content. In other words, some tags may not directly appear in the content; their relevant words appear instead. Likewise, the long-tailed distribution in Fig. 5 has a lower head and a higher tail which indicates a more significant tag-content relevance phenomenon.

Overall, based on the above studies, we can conclude that the tag-content relevance phenomenon exists in both data sets, although the occurrence degree may differ in different domains. Additionally, such occurrence largely exists whether we consider the relevant words or not, and it exists not only in the body of content but also in the title of the content. Intuitively, such phenomenon can help to find suitable tags for the given content. In the next sections, we will show how we use the tag-content relevance to boost the tag recommendation performance.

3. The proposed SimWord model

In this section, we present the proposed method. We start with the problem definition, and then discuss the SimWord model as well as some special cases.

3.1. Problem definition

Table 1 lists the main symbols used throughout this article. We use D to stand for a collection of input documents.[5] Each document d contains a list of words $\overrightarrow{W}_d$ and a list of tags $\overrightarrow{\Lambda}_d$. All the words form the vocabulary V, and all the tags form the tag space T. We denote K as the total number of latent topics in the input documents.

[5] In this article, we interchangeably use 'document' and 'post' as both of them indicate the online content that we aim to recommend tags for.

Table 1
Symbols.

Symbol	Definition and description
D	Collection of documents
V	Vocabulary
T	Tag space
M	Total number of documents
K	Total number of latent topics
$\vec{W}_d$	List of words in document d
$\vec{\Lambda}_d$	List of tags in document d
N_d	Number of words in document d

Table 2
Statistics of the datasets.

Dataset	# of posts	Vocab. size	# of tags	Avg words per post
Math	19,950	7705	461	54
SO	3,350,978	9357	1035	81
BibSonomy	247,889	7529	612	15
AskUbuntu	234,703	6714	743	82

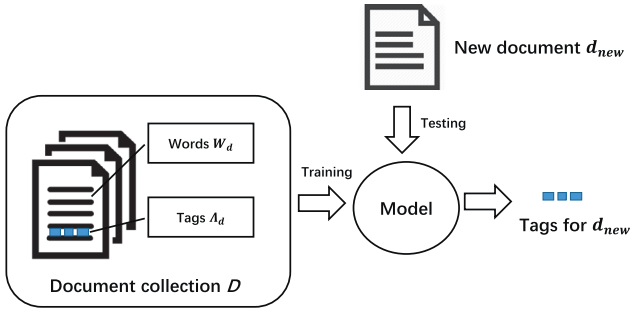


Fig. 6. The illustration of the tag recommendation problem.

With these notations, we define the tag recommendation problem as follows.

Problem 1. Tag recommendation problem

Given: (1) a collection of documents D, where each document $d \in D$ has its own words $\vec{W}_d$ and tags $\vec{\Lambda}_d$, and (2) a new document $d_{new} \notin D$ which only contains words $\vec{W}_{d_{new}}$;
Find: the estimated list of tags for the new document d_{new}.

An illustration of the above problem definition is shown in Fig. 6. As we can see, we assume that we have a collection of documents whose tags are already available. By training a model based on this document collection, we can predict the potential tags for a new document. Although not explicitly stated, tags in $\vec{\Lambda}_d$ as well as their relevant words may have appeared multiple times in $\vec{W}_d$ for a given document d. Explicitly identifying and exploiting such occurrence phenomenon is the main difference between our work and the existing work on this problem.

3.2. SimWord model

To solve the above problem, we build a supervised topic model SimWord to incorporate the tag-content relevance phenomenon. Existing supervised topic models learn a topic-word distribution for each tag/topic. In addition to generating words from topic-word distribution as existing methods do, SimWord tries to first construct relevant words for each tag, and directly generate a word from relevant words with a probability learned for different domains. Fig. 7 shows the graphical representation for the model.

- As we can see from the figure, SimWord builds on the LDA model [3] to generate words for documents. For each document, LDA assumes that it has several latent topics (θ). Words are generated from a specific topic (z) and the topic-word distributions (Φ).
- Then, following the LLDA model [25], we assume that the tags in Λ determine the latent topics during the generative process, and constrain that tag and latent topic are one-one correspondent. That is, each tag is associated with one topic, and the topic number of a document is the same with its tag number.[6]
- Then, to make use of the tag-content relevance, we further add a latent variable x to indicate the probability that the word w is generated by the tag-word distribution (Φ) or by the relevant-word distribution (Ω). Here, Ω contains the relevant words (including the tag itself) to the tag. The latent variable x is sampled from a Bernoulli distribution (Ψ), and it is dependent on the specific topic (z), i.e., different topics may have different probabilities.

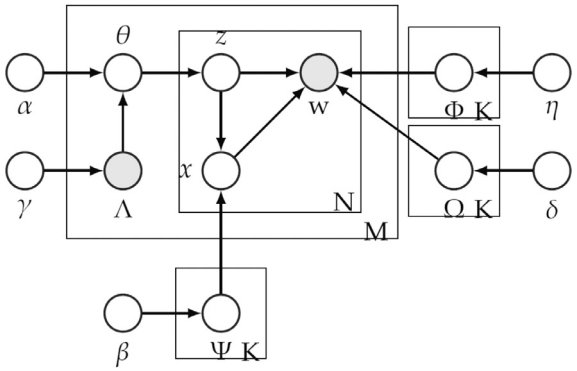


Fig. 7. Graphical representation for SimWord.

1 For each topic $k \in [1,K]$
2 Generate $\vec{\Phi}_k \sim Dir(\vec{\eta})$
3 Generate $\vec{\Omega}_k \sim Dir(\vec{\delta})$
4 Generate $\Psi_k \sim Beta(\vec{\beta})$
5 For each document $d \in [1,M]$
6 For each topic $k \in [1,K]$
7 Generate $\Lambda_{d,k} \in \{0,1\} \sim Bernoulli(\gamma_k)$
8 Generate $\vec{\alpha}_d = \vec{\Lambda}_d \circ \vec{\alpha}$
9 Generate $\vec{\theta}_d \sim Dir(\vec{\alpha}_d)$
10 For each word $i \in [1,N_d]$
11 Generate $z_i \sim Mult(\vec{\theta}_d)$
12 Generate $x_i \in \{0,1\} \sim Bernoulli(\Psi_{z_i})$
13 if $x_i = 1$
14 Generate $w_i \sim Mult(\vec{\Omega}_{z_i})$
15 else
16 Generate $w_i \sim Mult(\vec{\Phi}_{z_i})$

Fig. 8. Generative process of SimWord.

The generative process for our model is summarized in Fig. 8. For each topic k, Step 2 draws a multinomial topic-word distribution $\vec{\Phi}_k$ from a Dirichlet prior $\vec{\eta}$, Step 3 draws a multinomial relevant-word distribution $\vec{\Omega}_k$ from a Dirichlet prior $\vec{\delta}$, and Step 4 draws a Bernoulli distribution Ψ_k from a Beta prior $\vec{\beta}$. Here, Ψ_k indicates the probability to directly use the relevant words

[6] The total number of topics is the same with the total number of tags, i.e., $K = |T|$.

as the generated word for tag/topic k. For each document d, a multinomial distribution $\vec{\theta}_d$ is drawn over restricted topics that correspond to its tags $\vec{\Lambda}_d$[7] (Steps 6–9). In Step 8, we compute the Hardamard product between $\vec{\Lambda}_d$ and $\vec{\alpha}$, so that the topic assignment z_i for each word in document d is limited within its own tags. For each word i, we use a latent variable x_i to determine where it is generated from. When x_i equals 1, the word is generated from the multinomial distributions $\vec{\Omega}_{z_i}$ for the topic/tag (Steps 13–14). Otherwise, if x_i equals 0, the word is generated from the multinomial distributions $\vec{\Phi}_{z_i}$ for the topic/tag (Steps 15–16).

The key extension. The proposed SimWord is an extension from the LLDA model. From the topic modeling viewpoint, LLDA extends the LDA model by treating the tags as the supervision information of the corresponding content; SimWord further extends LLDA by explicitly specifying several relevant words for a given topic/tag, and allowing to generate the content directly using these words. By using the relevant words as guiding information, we may learn better combined topic/tag word distributions, and thus make more accurate recommendations based on the learned model.

Relevant word finding. For the above model, we need to first generate the relevant words for tags. There are some existing methods for this purpose such as WordNet and word2vec. In this work, since the vocabulary of word2vec is much larger, we choose to use it to generate relevant words.

Special Cases. Next, we discuss some special cases of our proposed method. There are several design choices of SimWord in terms of *(1) whether to use the relevant words or not* and *(2) whether to use the content body or not*. Some special cases are listed below.

- First, if we ignore the relevant words (i.e., use of the tags only), and if we only use the title as the content, we may have a special case SimWord$_t$.
- Next, if we ignore the relevant words, and use both title and body as content, we have the SimWord$_{tb}$ special case.
- Third, if we further incorporate relevant words, and only use the title as content, we have SimWord$_{ts}$.
- Finally, if we use relevant words, and use both title and body as content, we have SimWord$_{tbs}$ which is the proposed SimWord.

For the above special cases, using only title as content would potentially improve the training efficiency as we need to deal with a much smaller corpus. Whether using relevant words can help to validate the usefulness of taking relevant words into consideration. Other special cases such as ignoring the title and keeping the body in the content is of relatively low priority and thus are not studied in this work.

4. Model inference and prediction

In this section, we present the learning and prediction algorithms for the SimWord model.

4.1. Model inference

Here, we first describe the learning algorithm for the SimWord model. The algorithms for other special cases can be similarly obtained.

(A) Joint likelihood computation. We start with the computation of the following joint likelihood of the observed variables $\vec{W}$ and $\vec{\Lambda}$ and unobserved variables $\vec{Z}$ and $\vec{X}$, based on which we can derive update rules for θ, Ψ, Φ, and Ω.

$$p(\vec{Z},\vec{X},\vec{W},\vec{\Lambda}) = p(\vec{Z}|\vec{\alpha},\vec{\Lambda})p(\vec{X}|\vec{\beta},\vec{Z})p(\vec{\Lambda}|\vec{\gamma}) \\ p(\vec{W}|\vec{Z},\vec{X},\vec{\eta})p(\vec{W}|\vec{Z},\vec{X},\vec{\delta}). \quad (1)$$

For Eq. (1), considering the right side in sequence, we first have

$$\begin{aligned} p(\vec{Z}|\vec{\alpha},\vec{\Lambda}) &= \prod_{m=1}^{M} p(\vec{Z}_m|\vec{\alpha},\vec{\Lambda}) \\ &= \prod_{m=1}^{M}\int p(\vec{Z}_m|\vec{\Theta})p(\vec{\Theta}|\vec{\alpha},\vec{\Lambda})\,d\vec{\Theta} \\ &= \prod_{m=1}^{M}\int \prod_{k=1}^{K}\theta_{m,k}^{N_{m,k}}\frac{1}{\Delta(\vec{\alpha})}\prod_{k=1}^{K}\theta_{m,k}^{\alpha_{m,k}-1}\,d\vec{\Theta} \\ &= \prod_{m=1}^{M}\int \frac{1}{\Delta(\vec{\alpha})}\prod_{k=1}^{K}\theta_{m,k}^{N_{m,k}+\alpha_{m,k}-1}\,d\vec{\Theta} \\ &= \prod_{m=1}^{M}\frac{\Delta(\vec{N}_m+\vec{\alpha})}{\Delta(\vec{\alpha})}. \end{aligned} \quad (2)$$

where $\theta_{m,k}$ is the probability that document m belongs to topic k, $N_{m,k}$ is the number of words that belong to topic k in document m, and $\Delta(\vec{\alpha}) = \frac{\prod_{k=1}^{dim\,\vec{\alpha}}\Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{dim\,\vec{\alpha}}\alpha_k)}$.

Next, we turn to $p(\vec{X}|\vec{\beta},\vec{Z})$. By using similar expansion and integral techniques, we have

$$p(\vec{X}|\vec{\beta},\vec{Z}) = \prod_{k=1}^{K}\frac{B(\vec{N}_k+\vec{\beta})}{B(\vec{\beta})}. \quad (3)$$

where $\vec{X}$ is composed of 0s or 1s to determine where a word is generated from. In Eq. (3), we divide N_k into two parts: $N_{k,1}$ and $N_{k,0}$. The former one is the number of relevant words generated by topic/tag k, and the latter one indicates the number of words generated by topic-word distribution. $B(\vec{\beta})$ is a normalization constant to ensure that the total probability integrates to 1. $B(\vec{\beta})$ is defined as $B(\vec{\beta}) = \frac{\Gamma(\beta_1)\Gamma(\beta_0)}{\Gamma(\beta_1+\beta_0)}$.

Next, since the tags for each document are observed variables, the prior γ is unused. As a result, $p(\vec{\Lambda}|\vec{\gamma})$ is a constant and it can be ignored in the inference.

Finally, for $p(\vec{W}|\vec{Z},\vec{X},\vec{\eta})$ and $p(\vec{W}|\vec{Z},\vec{X},\vec{\delta})$, we have

$$p(\vec{W}|\vec{Z},\vec{X},\vec{\eta}) = \prod_{k=1}^{K}\frac{\Delta(\vec{N}_{k,0}+\vec{\eta})}{\Delta(\vec{\eta})}, \quad (4)$$

and

$$p(\vec{W}|\vec{Z},\vec{X},\vec{\delta}) = \prod_{k=1}^{K}\frac{\Delta(\vec{N}_{k,1}+\vec{\delta})}{\Delta(\vec{\delta})}, \quad (5)$$

where Eq. (4) only applies when $\vec{X}$ is set to 0, and Eq. (5) only applies when $\vec{X}$ is set to 1.

Putting the above equations together, we have

$$p(\vec{Z},\vec{X},\vec{W},\vec{\Lambda}) \propto \prod_{m=1}^{M}\frac{\Delta(\vec{N}_m+\vec{\alpha})}{\Delta(\vec{\alpha})}\cdot\prod_{k=1}^{K}\frac{B(\vec{N}_k+\vec{\beta})}{B(\vec{\beta})} \\ \cdot\prod_{k=1}^{K}\frac{\Delta(\vec{N}_{k,0}+\vec{\eta})}{\Delta(\vec{\eta})}\cdot\prod_{k=1}^{K}\frac{\Delta(\vec{N}_{k,1}+\vec{\delta})}{\Delta(\vec{\delta})}. \quad (6)$$

(B) CVB0 learning. Next, to solve the model, we can develop a Gibbs sampling algorithm to train the parameters. However, Gibbs

[7] The tags $\vec{\Lambda}_d$ in document d are observed variables in the model, and the prior γ is unused. We include it for completeness.

sampling is inherently stochastic and unstable when iterations are not enough. On the other hand, it is noticed that the CVB0 learning algorithm [1] converges faster and is more stable than the Gibbs sampling algorithm [23]. Therefore, we choose to build our learning algorithm based on CVB0 approximation algorithm.

For the proposed algorithm, we first need to obtain the Gibbs sampling rule for the unobserved variables $\vec{Z}$ and $\vec{X}$ in our model. We can apply Bayes rule to define the update formula for assigning a new z and x to a word i in document m as

$$p(Z_{m,i}=z, X_{m,i}=x|\vec{Z}_{m,-i}, \vec{X}_{m,-i}, \vec{W}, \vec{\Lambda})$$
$$= \frac{p(\vec{Z}_m, \vec{X}_m, \vec{W}, \vec{\Lambda})}{p(\vec{Z}_{m,-i}, \vec{X}_{m,-i}, \vec{W}, \vec{\Lambda})}$$
$$\propto \frac{p(\vec{Z}_m|\vec{\Lambda}, \vec{\alpha})p(\vec{X}_m|\vec{\beta}, \vec{Z}_m)p(\vec{W}|\vec{Z}_m, \vec{X}_m, \vec{\eta})p(\vec{W}|\vec{Z}_m, \vec{X}_m, \vec{\delta})}{p(\vec{Z}_{m,-i}|\vec{\Lambda}, \vec{\alpha})p(\vec{X}_{m,-i}|\vec{\beta}, \vec{Z}_{m,-i})p(\vec{W}_{-i}|\vec{Z}_{m,-i}, \vec{X}_{m,-i}, \vec{\eta})p(\vec{W}_{-i}|\vec{Z}_{m,-i}, \vec{X}_{m,-i}, \vec{\delta})}$$
$$\propto \begin{cases} \frac{N_{m,k,-i}+\alpha_k}{\sum_{k'=1}^{K}(N_{m,k',-i}+\alpha_{k'})} \cdot \frac{N_{k,1,-i}+\beta_1}{\sum_{x=0}^{1}(N_{k,x,-i}+\beta_x)} \cdot \frac{N_{k,v,1,-i}+\delta_v}{\sum_{v'=1}^{V}(N_{k,v',1,-i}+\delta_{v'})}, & \text{k=w,x=1} \\ \frac{N_{m,k,-i}+\alpha_k}{\sum_{k'=1}^{K}(N_{m,k',-i}+\alpha_{k'})} \cdot \frac{N_{k,0,-i}+\beta_0}{\sum_{x=0}^{1}(N_{k,x,-i}+\beta_x)} \cdot \frac{N_{k,v,0,-i}+\eta_v}{\sum_{v'=1}^{V}(N_{k,v',0,-i}+\eta_{v'})}, & \text{k=w,x=0} \\ \frac{N_{m,k,-i}+\alpha_k}{\sum_{k'=1}^{K}(N_{m,k',-i}+\alpha_{k'})} \cdot \frac{N_{k,v,0,-i}+\eta_v}{\sum_{v'=1}^{V}(N_{k,v',0,-i}+\eta_{v'})}, & \text{k} \neq \text{w} \end{cases} \quad (7)$$

where $N_{m,k,-i}$ indicates the number of words that belong to tag/topic k in document m when the current word is excluded, $N_{k,1,-i}$ indicates the number of relevant words that belong to tag/topic k and are directly generated by k from relevant-word distribution when the current word is excluded, and $N_{k,v,1,-i}$ indicates the number of relevant word v that belongs to tag/topic k and is generated by the relevant-word distribution when the current word is excluded. $N_{k,0,-i}$ indicates the number of words that belong to tag/topic k and are generated by the tag-word distribution when the current word is excluded, and $N_{k,v,0,-i}$ indicates the number of word v that belongs to tag/topic k and is generated by the tag-word distribution when the current word is excluded. For the above rule, when $k=w, x=1$, it means that the word w is generated by tag k directly from relevant-word distribution, and the probability is decided by θ, Ψ, and Ω. On the other side, when $k \neq w$, the probability only includes θ and Φ.

Based on the sampling rule in Eq. (7), we can build our learning algorithm. For a word i in document m, we store the probabilities of each assignment z and x. Thus, we can have a distribution $\gamma_{m,i}$ over the likelihood that the word is generated by each (z, x) pair using the Gibbs sampling rule (Eq. (7)). Then, we can derive the update rules based on CVB0. For the tag k who has a relevant word i, we have

$$\gamma_{m,i,k,1} \propto (n_{m,k,-i}+\alpha_k) \cdot \frac{n_{k,1,-i}+\beta_1}{\sum_{x=0}^{1}(n_{k,x,-i}+\beta_x)} \cdot \frac{n_{k,v,1,-i}+\eta_v}{\sum_{v'=1}^{V}(n_{k,v',1,-i}+\eta_{v'})}$$
$$\gamma_{m,i,k,0} \propto (n_{m,k,-i}+\alpha_k) \cdot \frac{n_{k,0,-i}+\beta_0}{\sum_{x=0}^{1}(n_{k,x,-i}+\beta_x)} \cdot \frac{n_{k,v,0,-i}+\eta_v}{\sum_{v'=1}^{V}(n_{k,v',0,-i}+\eta_{v'})} \quad (8)$$

and otherwise, we have

$$\gamma_{m,i,k} \propto (n_{m,k,-i}+\alpha_k) \cdot \frac{n_{k,v,0,-i}+\eta_v}{\sum_{v'=1}^{V}(n_{k,v',0,-i}+\eta_{v'})} \quad (9)$$

The counting variables are defined as

$$n_{m,k,-i} = \sum_w \sum_x \gamma_{m,w,k,x} - \gamma_{m,i,k}$$
$$n_{k,1,-i} = \sum_d \sum_w \gamma_{d,w,k,1} - \gamma_{m,i,k,1} \quad n_{k,v,1,-i} = \sum_d \gamma_{d,v,k,1} - \gamma_{m,i,k,1}$$
$$n_{k,0,-i} = \sum_d \sum_w \gamma_{d,w,k,0} - \gamma_{m,i,k,0} \quad n_{k,v,0,-i} = \sum_d \gamma_{d,v,k,0} - \gamma_{m,i,k,0} \quad (10)$$

All the counting variables on the right side of the above equations refer to the values of the previous iteration. At the very beginning, we can take small random values for γ into the equations to initialize these counting variables.

The purpose of the training process is to obtain θ, Ψ, Φ, and Ω, where θ represents the topic distribution of each document, Ψ represents the distribution to indicate whether the word is generated by the relevant-word distribution or generated by the topic-word distribution, Φ represents the word distribution of each topic and Ω represents the relevant-word distribution of each topic. The formulas for these parameters are listed as follows.

$$\theta = \frac{n_{m,k}+\alpha_k}{\sum_{k'=1}^{K}(n_{m,k'}+\alpha_{k'})} \qquad \Psi = \frac{n_{k,1}+\beta_1}{\sum_{x=0}^{1}(n_{k,x}+\beta_x)}$$
$$\Phi = \frac{n_{k,v,0}+\eta_v}{\sum_{v=1}^{V}(n_{k,v,0}+\eta_v)} \qquad \Omega = \frac{n_{k,v,1}+\delta_v}{\sum_{v=1}^{V}(n_{k,v,1}+\delta_v)} \quad (11)$$

Finally, the SimWord algorithm is summarized in Algorithm. 1.

Algorithm 1 The SimWord Algorithm.

Input: D: Collection of Documents
Output: θ, Ψ, Φ, Ω

```
1:  for document m ← 1, M do
2:      for word i ← 1, N do
3:          for tag k ← 1, K do
4:              if word i in tag k's relevant words then
5:                  γ_{m,i,k,1} ← random(); n_{k,1} ← n_{k,1} + γ_{m,i,k,1}; n_{k,v,1} ← n_{k,v,1} + γ_{m,i,k,1}
6:              end if
7:              γ_{m,i,k,0} ← random(); n_{k,0} ← n_{k,0} + γ_{m,i,k,0}; n_{k,v,0} ← n_{k,v,0} +
                γ_{m,i,k,0}; n_{m,k} ← n_{m,k} + γ_{m,i,k}
8:          end for
9:      end for
10: end for
11: repeat
12:     for document m ← 1, M do
13:         for word i ← 1, N do
14:             for tag k ← 1, K do
15:                 if word i in tag k's relevant words then
16:                     n_{k,1} ← n_{k,1} − γ_{m,i,k,1}; n_{k,v,1} ← n_{k,v,1} − γ_{m,i,k,1}
17:                 end if
18:                 n_{k,0} ← n_{k,0} − γ_{m,i,k,0}; n_{k,v,0} ← n_{k,v,0} − γ_{m,i,k,0}; n_{m,k} ← n_{m,k} − γ_{m,i,k}
19:                 if word i in tag k's relevant words then
20:                     update γ_{m,i,k} ← Eq. (8)
21:                 else
22:                     update γ_{m,i,k} ← Eq. (9)
23:                 end if
24:                 if word i in tag k's relevant words then
25:                     n_{k,1} ← n_{k,1} + γ_{m,i,k,1}; n_{k,v,1} ← n_{k,v,1} + γ_{m,i,k,1}
26:                 end if
27:                 n_{k,0} ← n_{k,0} + γ_{m,i,k,0}; n_{k,v,0} ← n_{k,v,0} + γ_{m,i,k,0}; n_{m,k} ← n_{m,k} + γ_{m,i,k}
28:             end for
29:         end for
30:     end for
31: until Iterations ≥ MaxIterations
32: compute θ, Ψ, Φ, Ω via Eq. (11)
33: return θ, Ψ, Φ, Ω
```

The algorithm takes the collection of documents as input and outputs four parameters. Line 1–10 initialize all the γ for each word with small random values, and then assign values of counting variables separately. Line 11–31 iteratively estimate the parameters based on CVB0 approximation. Line 16 and 18 exclude the current assignment of γ from those counting variables. Line 20 and 22 are the core update steps in the iteration. For each tag k, we use Eq. (8) or Eq. (9) to estimate the probability that tag k will be assigned for word i in document m. After updating γ, we reassign the counting variables with new γ in Line 25 and 27. The iterative process terminates when the parameters converge or when the maximum iteration number is reached.

4.2. Algorithm analysis

Here, we briefly analyze the time complexity of the SimWord algorithm. Basically, SimWord scales linearly w.r.t the data size as summarized in the following lemma. We will also experimentally validate this in the next section.

Lemma 1. *The time complexity of Algorithm 1 is $O(WKl)$, where W is the vocabulary size, K is the number of tags, and l is the maximum iteration number.*

Proof. The initial process (line 1–10) in Algorithm 1 requires $O(\sum_i^M N_i K)$ time where N_i is the number of words in the ith document and K is the number of tags. The main iterations begin from line 11 to 31, and the time cost is $O(\sum_i^M N_i K l)$, where l is the number of maximum iterations. Rewriting $\sum_i^M N_i$ as the total number of words in documents, the overall time complexity of the proposed algorithm is $O(WKl)$. □

4.3. Prediction

Here, we explain how to use the learned model to predict tags for a new post. To finish this task, we need to obtain the posterior tag distribution for each post using only content as input. The posterior is

$$p(t|d_{new}) = \sum_w p(t|w)p(w|d_{new}) = \sum_w \frac{p(w|t)p(t)}{p(w)} p(w|d_{new}). \tag{12}$$

By assuming that the probability $p(t)$ can be inferred approximately through the training set, we have

$$p(t) = \sum_d p(t|d)p(d). \tag{13}$$

Here, we assume that the appearing chance of each post is the same, and thus we can ignore the $p(d)$ term in the above equation. For $p(t|d)$, we directly use the θ learned by our model.

Next, since each word w is generated from a certain topic t, the probability $p(w)$ can be computed as

$$p(w) = \sum_t p(w|t)p(t). \tag{14}$$

To compute Eq. (12), we now only need to compute $p(w|t)$ and $p(w|d_{new})$. We will first compute $p(w|d_{new})$, and come back to the computation of $p(w|t)$ later.

For a new post, $p(w|d_{new})$ can be calculated as

$$p(w|d_{new}) = \frac{count(w)}{len(d_{new})}, \tag{15}$$

where $count(w)$ means the number of word w and $len(d_{new})$ means the total number of words in document d_{new}.

Finally, for $p(w|t)$, the computation is as follows

$$\begin{aligned} p(w|t) &= \begin{cases} p(w|t,x=0)p(x=0|w\sim t) \\ +p(w|t,x=1)p(x=1|w\sim t), & \text{w is a relevant word of t} \\ p(w|t,x=0)p(x=0|w\nsim t), & \text{o.w.} \end{cases} \\ &= \begin{cases} \Phi\cdot(1-\Psi)+\Omega\cdot\Psi, & \text{w is a relevant word of t} \\ \Phi, & \text{o.w.} \end{cases}, \end{aligned} \tag{16}$$

where we consider $p(w|t)$ in two circumstances. When the word is generated from the relevant-word distribution, $p(w|t,x=1)$ is estimated by Ω. Similarly, when the word is generated from the tag-word distribution, $p(x=0|w\neq t)$ is a constant 1. Then, we use Φ as a substitute for $p(w|t,x=0)$ and Ψ as an approximation of $p(x=1|w\sim t)$. Here, $w\sim t$ means that word w is a relevant word of tag t while $w\nsim t$ means otherwise.

5. Experimental evaluations

In this section, we present our experimental evaluations. The experiments are designed to answer the following questions:

- *Effectiveness*: How accurate is the proposed algorithm for tag recommendation?
- *Efficiency*: How scalable is the proposed algorithm?

5.1. Experimental setup

In addition to the two data sets described in Section 2, we further study the BibSonomy[8] and AskUbuntu[9] data sets. For the BibSonomy data, it only contains the post body (no post title). For the AskUbuntu data, it contains both post title and post body. These two additional datasets are also publicly available for research purpose. Similar preprocessing steps are applied on the four datasets (see Section 2), and Table 2 shows the statistics of the four preprocessed data sets.

For all the studied four data sets, we randomly select 90% posts as training data and use the rest as test data. Since some compared methods are computationally prohibitive on the whole SO data, we also randomly sample two subsets (SO-10K with 10,000 posts and SO-100K with 100,000 posts) of the SO data to compare their effectiveness results. For the test set, we sort the tags based on the estimated probability ($p(t|d_{new})$) in the descending order, and use the ranked list as output. For the hyper-parameters, we fix α, η, δ, and β to $50/K$, 0.01, 0.01, and 0.1, respectively.

As to evaluation metrics, we adopt Recall@n for effectiveness comparison. The reason is that finding all the useful tags in the recommendation list is important for tag recommendation tasks [27]. As to the list size n, we choose $n=5$ and $n=10$ as such choices will not cause many burdens to the users. Recall@n is defined as follows

$$Recall@n = \frac{1}{M}\sum_{i=1}^{M}\frac{hit(n)_i}{tag_i}, \tag{17}$$

where M is the number of posts in the data, $hit(n)_i$ is the number of tags that have been successfully recommended in the top-n ranked list, and tag_i is the number of actual tags of the ith post.

For efficiency, we simply report the wall-clock time of the proposed algorithm. All the experiments were run on a machine with eight 3.4 GHz Intel Cores and 32 GB memory.

5.2. Experiemental results

Next, we present the experimental results including the effectiveness and efficiency results, as well as their tradeoffs.

5.2.1. Effectiveness results

(A) Effectiveness comparisons. For effectiveness, we first compare the proposed SimWord with several existing methods including LLDA [25], Link-LDA [4], MATAR [13], Snaff [14], and Maxide [44]. In the compared methods, LLDA and Link-LDA are topic models, MATAR and Maxide are multi-label learning methods, and Snaff is a hybrid method that combines several simple recommenders. The proposed SimWord belongs to the topic models. The results of Recall@5 and Recall@10 on all the data sets are shown in Tables 3 and 4, respectively. For symbols "–", we do not show the results (of Maxide and MATAR) as they are computationally prohibitive (e.g., cannot return results in 24 h on the corresponding data).

There are several observations from the tables. First, SimWord outperforms all the compared methods on all the data sets. For

[8] https://www.bibsonomy.org/
[9] http://askubuntu.com/

Table 3
The Recall@5 results on all the data sets. Higher is better. "–" means that the corresponding method is computationally prohibitive on the corresponding data. SimWord outperforms all the compared methods.

Data	LLDA	Link-LDA	MATAR	Snaff	Maxide	SimWord
SO-10K	0.47805	0.33651	0.48988	0.38248	0.38995	**0.57365**
SO-100K	0.54254	0.32907	0.53774	0.45165	0.50998	**0.62271**
SO	0.55660	0.33847	–	0.46402	–	**0.63311**
Math	0.58859	0.43850	0.55271	0.48833	0.52436	**0.62337**
BibSonomy	0.43473	0.19169	–	0.38523	0.41898	**0.45794**
AskUbuntu	0.56973	0.28009	–	0.45250	0.57035	**0.65077**

Table 4
The Recall@10 results on all the data sets. Higher is better. "–" means that the corresponding method is computationally prohibitive on the corresponding data. SimWord outperforms all the compared methods.

Data	LLDA	Link-LDA	MATAR	Snaff	Maxide	SimWord
SO-10K	0.58770	0.43010	0.55168	0.48103	0.46815	**0.68358**
SO-100K	0.64804	0.42514	0.59125	0.54921	0.62361	**0.70638**
SO	0.66422	0.43429	–	0.58363	–	**0.71379**
Math	0.68992	0.58259	0.65750	0.57042	0.64942	**0.72647**
BibSonomy	0.53767	0.27113	–	0.48924	0.51487	**0.55012**
AskUbuntu	0.69983	0.40765	–	0.55016	0.70772	**0.76761**

Table 5
The Recall@5 results of SimWord and its special cases. "–" is due to the fact that the BibSonomy data does not contain titles.

Data	SimWord	SimWord_*ts*	SimWord_*tb*	SimWord_*t*
SO-10K	0.57365	**0.59548**	0.56330	0.58538
SO-100K	0.62271	**0.65037**	0.60939	0.63607
SO	0.63311	**0.65785**	0.62169	0.64548
Math	**0.62337**	0.61358	0.61057	0.59352
BibSonomy	**0.45794**	–	0.44702	–
AskUbuntu	**0.65077**	0.61117	0.63561	0.60790

Table 6
The Recall@10 results of SimWord and its special cases. "–" is due to the fact that the BibSonomy data does not contain titles.

Data	SimWord	SimWord_*ts*	SimWord_*tb*	SimWord_*t*
SO-10K	**0.68358**	0.66035	0.67707	0.65870
SO-100K	0.70638	**0.71953**	0.70339	0.71477
SO	0.71379	**0.73204**	0.71036	0.72682
Math	**0.72647**	0.70106	0.71797	0.68529
BibSonomy	**0.55012**	–	0.54901	–
AskUbuntu	**0.76761**	0.70221	0.75777	0.69918

example, on SO-10K data, SimWord improves its best competitors (MATAR and LLDA) by 17.1% w.r.t Recall@5 and by 16.3% w.r.t Recall@10. On Math data, SimWord is 5.9% and 5.3% better than the best competitor LLDA for Recall@5 and Recall@10, respectively. Similarly, SimWord outperforms its best competitor LLDA by 5.3% w.r.t Recall@5 and 2.3% w.r.t Recall@10 on the BibSonomy data, and it is 14.1% and 8.5% better than its best competitor Maxide for Recall@5 and Recall@10 on the AskUbuntu data. Actually, LLDA can be seen a special case of SimWord if we do not consider the tag-content relevance phenomenon in the model. This result indicates that the tag-content relevance indeed can help improve the recommendation accuracy. Second, we find that the improvement of SimWord are more significant on the SO data than the Math data. This is probably due to the fact that the tag-content relevance degree is higher on SO than that on Math (as indicated by the empirical study in Section 2).

(B) Effectiveness of special cases. Next, we study the special cases of SimWord including SimWord$_{ts}$, SimWord$_{tb}$, and SimWord$_{t}$. The results are shown in Tables 5 and 6. We do not report some of the results on BibSonomy due to the fact that the BibSonomy data does not contain titles while the corresponding methods are special cases that use title only as input.

First, we can observe from the tables that SimWord has better performance than the SimWord$_{tb}$ special case in general. For example, on the Math data, SimWord improves SimWord$_{tb}$ by 2.1% and 1.2% in terms of Recall@5 and Recall@10, respectively. Since SimWord can be seen as adding relevant words into SimWord$_{tb}$, this result further validates the usefulness of taking relevant words (in addition to the tags) into consideration. Next, SimWord$_{ts}$ can already achieve good performance (e.g., it is even better than SimWord on the SO data). This is due to the fact that even when only the title is considered, the tag-content relevance degree is already very high on the SO data while the word vocabulary is much smaller. This result means that we can recommend tags solely based on the title of the content, if there are plenty tags appeared in the title.

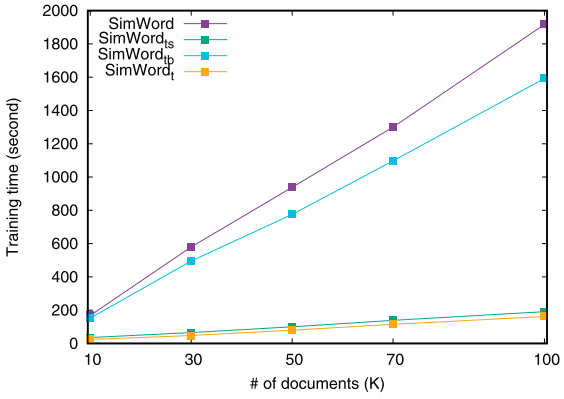


Fig. 9. The scalability of the proposed SimWord algorithm. SimWord as well as its special cases scales linearly w.r.t the data size.

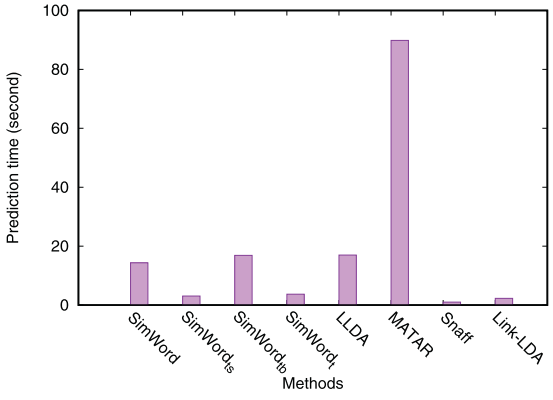
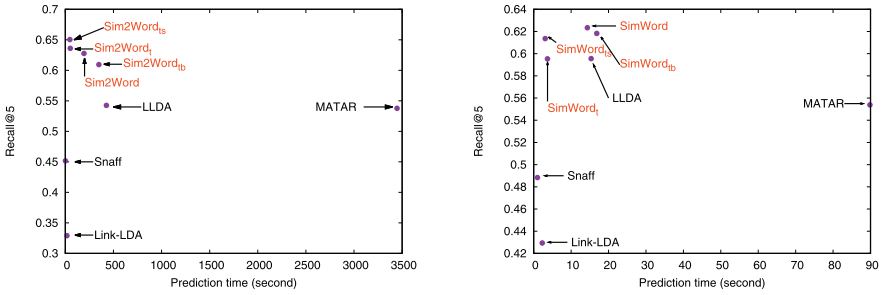


Fig. 10. Response time comparison. SimWord and its special cases have a favorable response speed.

5.2.2. Efficiency results

(C) Scalability. Next, we study the scalability of the proposed algorithm in the training stage. For scalability, we are interested in how the wall-clock time grows as the number of documents increases. Therefore, we vary the size of the training data on the SO data by randomly sampling the subsets of documents, and report the results of wall-clock time in Fig. 9. Similar results are observed on the other data sets, and we omit the figures for brevity.

As we can see from the figure, SimWord and its special cases scale linearly w.r.t the size of training data (the number of posts), which is also consistent with our algorithm analysis in Section 4.2. Additionally, SimWord$_{ts}$ runs much faster than SimWord (around 9x faster), as it only involves the title as input.



(a) SO-100K (b) Math

Fig. 11. The quality-speed tradeoffs. SimWord and its special cases achieve a good balance.

(D) Response time. We also compare the prediction time for a new post at the response stage of different methods. The results on Math data are showed in Fig. 10. Similar results are observed on the other data sets. Here, we do not show the result of Maxide as its response stage is mixed with the training stage. As we can see, SimWord and its special cases can make predictions within 20 s. SimWord and LLDA have comparable response speed. The response time of MATAR is long as it adopts the lazy strategy. Snaff and Link-LDA are faster, and Sim2Word$_{ts}$ and Sim2Word$_t$ can have close response time with them.

5.2.3. Quality-speed tradeoff

(E) Quality-speed tradeoffs. Finally, we study the quality-speed tradeoffs of different methods. The results are shown in Fig. 11. In the figures, x-axis indicates the wall-clock time used at the response stage, and y-axis indicates the recommendation accuracy (Recall@5). Ideally, we want an algorithm sitting in the left-top corner. As we can see, our SimWord and special cases are all in the left-top corner. For example, compared with LLDA, SimWord$_{ts}$ is 8.0x faster w.r.t wall-clock time and 19.9% better w.r.t recall@5 on the SO-100K data. Although Snaff and Link-LDA are faster than our methods, the accuracy of these two methods are much worse. Overall, our methods achieve a good balance between the recommendation accuracy and the efficiency.

6. Related work

In this section, we briefly review the related work. We roughly divide existing tag recommendation methods into collaborative filtering method and content-based method.

The key insight of collaborative filtering method is to employ the tagging histories (i.e., user-item-tag tuples). For example, Symeonidis et al. [34] model users, items, and tags into 3-order tensors and use high-order singular value decomposition to recommend tags; Rendle et al. [27,28] further model the pairwise rankings into tensor factorization; Fang et al. [5] propose a non-linear tensor factorization method via Gaussian kernel; Feng and Wang [6] model a social tagging system as a multi-type graph, and recommend tags by learning the weights of nodes and edges in the graph; Zhao et al. [46] model the relationships in tagging data as a heterogeneous graph and propose a ranking algorithmic framework on the heterogeneous graph for tag recommendation. Other examples in this category include [9,10,16,31,32,47]. Methods in this class are more suitable to recommend a list of personalized tags for a fixed set of items, and there may be difficulties for them to recommend tags for new content.

Our method falls into the category of content based method. In contrast to collaborative filtering method, content based method takes the content as input, and therefore could be used to recommend tags for new content. For example, Sood et al. [33] and Mishne [20] leverage previous tags associated with similar content to recommend tags for new content. Murfi and Obermayer [21] first use keyword extraction to filter candidate tags and then apply non-negative matrix factorization for tag recommendation; Pudota et al. [24] combine keywords extraction with domain ontologies to recommend tags; Wang et al. [38] also extract keywords, and then apply association rules to recommend tags; Erosheva et al. [4] extend LDA by mixing the generation of tags and words; Ramage et al. [25] also extend LDA by constraining the one-one correspondence between tags and latent topics; Wang et al. [41] consider the cases when tags and classes are both available, and propose to fuse the tag recommendation task and the classification task together to improve the performance of each other. Other examples include [11,12,26,29,30,40]. Our method falls into this category of content based method. Different from the above work, our main observation is the tag-content relevance phenomenon, and we propose a generative model to integrates this phenomenon.

There also exists related work that combines the collaborative filtering and content based methods. For example, Lops et al. [17] and Wang et al. [36,37] design hybrid tag recommenders that combine collaborative filtering with content modeling. However, these methods still suffer from the cold-start problem of collaborative filtering methods; i.e., it is difficult for them to recommend suitable tags for new content.

Recently, there are also other lines of research that recommend tags for different types of content (e.g., tag recommendation for images [8,22,45]). For example, Liu et al. [15] explore locations to recommend tags for photos. Toderici et al. [35] present a system that automatically recommends tags for YouTube videos solely based on their audiovisual content. Font et al. [7] propose a tag recommendation system in an online platform for audio clip sharing. As to textual content, Xia et al. [43] and Wang et al. [39] provide tagging methods for software information sites. Basically, they combine several existing methods as subcomponents (e.g., multi-label prediction, TF-IDF similarity, LLDA, etc.), while we propose a new, unified model for content based tag recommendation.

Finally, this work is an extension of our previous conference paper [42] which can be seen as a special case of SimWord. That is, the previous work considers the case when the tags exactly appear in the content, while we further consider the case where the synonymous words of tags may also appear in the content.

7. Conclusions

In this article, we have proposed a content based tag recommendation model SimWord. The basic idea of SimWord is to make use of the tag-content relevance phenomenon that we have empirically verified. Some special cases of SimWord are also studied. Experimental evaluations on four real data sets show that the proposed methods can lead up to 17.1% improvement over the

best competitors in terms of prediction accuracy, while enjoying linear scalability in the training stage. We believe that considering the co-occurrences of tags and content words is a key building block for the tag recommendation problem on textual content.

There are several future directions. First, we adopt topic modeling as a tool to capture the semantics of textual content in this work; in the future, we consider to apply deep learning models such as convolutional neural networks and recurrent neural networks to see their effectiveness. Second, we remove some low frequency tags in our experimental setting; we are interested in checking the effectiveness of the existing methods for recommending these low frequency tags. Finally, the tags for the same piece of content may be correlated with each other; we plan to incorporate this phenomenon in the future work.

Acknowledgement

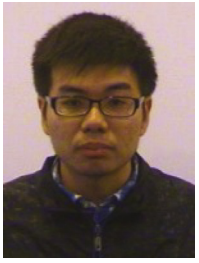
This work is supported by the National Key R&D Program of China (No. 2016YFB1000802), the National Natural Science Foundation of China (No. 61690204, 61672274, 61702252), and the Collaborative Innovation Center of Novel Software Technology and Industrialization. Hanghang Tong is partially supported by NSF (IIS-1651203, IIS-1715385, CNS-1629888 and IIS-1743040), DTRA (HDTRA1-16-0017), ARO (W911NF-16-1-0168), and gifts from Huawei and Baidu.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.neucom.2018.07.011.

References

[1] A. Asuncion, M. Welling, P. Smyth, Y.W. Teh, On smoothing and inference for topic models, in: Proceedings of the UAI, 2009, pp. 27–34.
[2] F.M. Belém, J.M. Almeida, M.A. Gonçalves, A survey on tag recommendation methods, J. Assoc. Inf. Sci. Technol. 68 (4) (2017) 830–844.
[3] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.
[4] E. Erosheva, S. Fienberg, J. Lafferty, Mixed-membership models of scientific publications, Proc. Natl. Acad. Sci. 101 (suppl 1) (2004) 5220–5227.
[5] X. Fang, R. Pan, G. Cao, X. He, W. Dai, Personalized tag recommendation through nonlinear tensor factorization using gaussian kernel, in: Proceedings of the AAAI, 2015.
[6] W. Feng, J. Wang, Incorporating heterogeneous information for personalized tag recommendation in social tagging systems, in: Proceedings of the KDD, 2012, pp. 1276–1284.
[7] F. Font, J. Serra, X. Serra, Class-based tag recommendation and user-based evaluation in online audio clip sharing, Knowl. Sys. 67 (2014) 131–142.
[8] Y. Gong, Q. Zhang, Hashtag recommendation using attention-based convolutional neural network, in: Proceedings of the IJCAI, 2016, pp. 2782–2788.
[9] S. Hamouda, N. Wanas, PUT-tag: personalized user-centric tag recommendation for social bookmarking systems, Soc. Netw. Anal. Min. 1 (4) (2011) 377–385.
[10] R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, G. Stumme, Tag recommendations in social bookmarking systems, AI Commun. 21 (4) (2008) 231–247.
[11] R. Krestel, P. Fankhauser, Personalized topic-based tag recommendation, Neurocomputing 76 (1) (2012) 61–70.
[12] R. Krestel, P. Fankhauser, W. Nejdl, Latent Dirichlet allocation for tag recommendation, in: Proceedings of the RecSys, 2009, pp. 61–68.
[13] L. Li, Y. Yao, F. Xu, J. Lu, MATAR: keywords enhanced multi-label learning for tag recommendation, in: Proceedings of the APWeb, 2015, pp. 268–279.
[14] M. Lipczak, E. Milios, Learning in efficient tag recommendation, in: Proceedings of the RecSys, 2010, pp. 167–174.
[15] J. Liu, Z. Li, J. Tang, Y. Jiang, H. Lu, Personalized Geo-specific tag recommendation for photos on social websites, IEEE Trans. Multimed. 16 (3) (2014) 588–600.
[16] R. Liu, Z. Niu, A collaborative filtering recommendation algorithm based on tag clustering, in: Future Information Technology, Springer, 2014, pp. 177–183.
[17] P. Lops, M. De Gemmis, G. Semeraro, C. Musto, F. Narducci, Content-based and collaborative techniques for tag recommendation: an empirical evaluation, J. Intell. Inf. Sys. 40 (1) (2013) 41–61.
[18] Y.-T. Lu, S.-I. Yu, T.-C. Chang, J.Y.-j. Hsu, A content-based method to enhance tag recommendation, in: Proceedings of the IJCAI, 9, 2009, pp. 2064–2069.
[19] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv:1301.3781 (2013).
[20] G. Mishne, AutoTag: a collaborative approach to automated tag assignment for weblog posts, in: Proceedings of the WWW, 2006, pp. 953–954.
[21] H. Murfi, K. Obermayer, A two-level learning hierarchy of concept based keyword extraction for tag recommendations, in: Proceedings of the ECML PKDD Discovery Challenge, 2009, pp. 201–214.
[22] H.T.H. Nguyen, M. Wistuba, L. Schmidt-Thieme, Personalized tag recommendation for images using deep transfer learning, in: Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2017, pp. 705–720.
[23] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, M. Welling, Fast collapsed Gibbs sampling for latent Dirichlet allocation, in: Proceedings of the KDD, 2008, pp. 569–577.
[24] N. Pudota, A. Dattolo, A. Baruzzo, F. Ferrara, C. Tasso, Automatic keyphrase extraction and ontology mining for content-based tag recommendation, Int. J. Intell. Sys. 25 (12) (2010) 1158–1186.
[25] D. Ramage, D. Hall, R. Nallapati, C.D. Manning, Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora, in: Proceedings of the EMNLP, 2009, pp. 248–256.
[26] D. Ramage, C.D. Manning, S. Dumais, Partially labeled topic models for interpretable text mining, in: Proceedings of the KDD, 2011, pp. 457–465.
[27] S. Rendle, L. Balby Marinho, A. Nanopoulos, L. Schmidt-Thieme, Learning optimal ranking with tensor factorization for tag recommendation, in: Proceedings of the KDD, 2009, pp. 727–736.
[28] S. Rendle, L. Schmidt-Thieme, Pairwise interaction tensor factorization for personalized tag recommendation, in: Proceedings of the WSDM, 2010, pp. 81–90.
[29] A.K. Saha, R.K. Saha, K.A. Schneider, A discriminative model approach for suggesting tags automatically for stack overflow questions, in: Proceedings of the Tenth Working Conference on Mining Software Repositories, IEEE Press, 2013, pp. 73–76.
[30] P. Seitlinger, D. Kowald, C. Trattner, T. Ley, Recommending tags with a model of human categorization, in: Proceedings of the CIKM, 2013, pp. 2381–2386.
[31] B. Sigurbjörnsson, R. Van Zwol, Flickr tag recommendation based on collective knowledge, in: Proceedings of the WWW, 2008, pp. 327–336.
[32] Y. Song, L. Zhang, C.L. Giles, Automatic tag recommendation algorithms for social recommender systems, ACM Trans. Web (TWEB) 5 (1) (2011) 4.
[33] S. Sood, S. Owsley, K.J. Hammond, L. Birnbaum, TagAssist: automatic tag suggestion for blog posts, in: Proceedings of the ICWSM, 2007.
[34] P. Symeonidis, A. Nanopoulos, Y. Manolopoulos, Tag recommendations based on tensor dimensionality reduction, in: Proceedings of the RecSys, 2008, pp. 43–50.
[35] G. Toderici, H. Aradhye, M. Pasca, L. Sbaiz, J. Yagnik, Finding meaning on YouTube: tag recommendation and category discovery, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 3447–3454.
[36] H. Wang, B. Chen, W.-J. Li, Collaborative topic regression with social regularization for tag recommendation, in: Proceedings of the IJCAI, 2013, pp. 2719–2725.
[37] H. Wang, X. Shi, D.-Y. Yeung, Relational stacked denoising autoencoder for tag recommendation, in: Proceedings of the AAAI, 2015, pp. 3052–3058.
[38] J. Wang, L. Hong, B.D. Davison, Tag recommendation using keywords and association rules, in: Proceedings of the KDD, 2009.
[39] S. Wang, D. Lo, B. Vasilescu, A. Serebrenik, EnTagRec: an enhanced tag recommendation system for software information sites, in: Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), 2014, pp. 291–300.
[40] T. Wang, H. Wang, G. Yin, C.X. Ling, X. Li, P. Zou, Tag recommendation for open source software, Front. Comput. Sci. 8 (1) (2014) 69–82.
[41] Y. Wang, S. Wang, J. Tang, G.-J. Qi, H. Liu, B. Li, CLARE: a joint approach to label classification and tag recommendation, in: Proceedings of the AAAI, 2017, pp. 210–216.
[42] Y. Wu, Y. Yao, F. Xu, H. Tong, J. Lu, Tag2Word: using tags to generate words for content based tag recommendation, in: Proceedings of the Twenty-fifth ACM International on Conference on Information and Knowledge Management, ACM, 2016, pp. 2287–2292.
[43] X. Xia, D. Lo, X. Wang, B. Zhou, Tag recommendation in software information sites, in: Proceedings of the Tenth Working Conference on Mining Software Repositories, IEEE Press, 2013, pp. 287–296.
[44] M. Xu, R. Jin, Z.-H. Zhou, Speedup matrix completion with side information: application to multi-label learning, in: Proceedings of the NIPS, 2013, pp. 2301–2309.
[45] Q. Zhang, J. Wang, H. Huang, X. Huang, Y. Gong, Hashtag recommendation for multimodal microblog using co-attention network, in: Proceedings of the Twenty-sixth International Joint Conference on Artificial Intelligence, 2017, pp. 3420–3426.
[46] W. Zhao, Z. Guan, Z. Liu, Ranking on heterogeneous manifolds for tag recommendation in social tagging services, Neurocomputing 148 (2015) 521–534.
[47] N. Zheng, Q. Li, A recommender system based on tag and time information for social tagging systems, Expert Syst. Appl. 38 (4) (2011) 4575–4587.

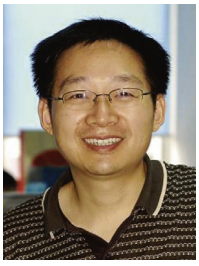


Yong Wu is a master student in the Department of Computer Science and Technology, Nanjing University, China. His research intersts include tag recommendation and probabilistic models.

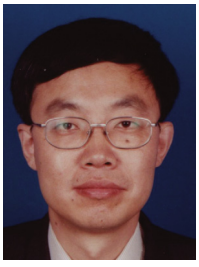
Shengqu Xi is a Ph.D. student in the Department of Computer Science and Technology, Nanjing University, China. His research intersts include software intelligence and recurrent neural networks.

Yuan Yao is currently an assistant researcher in the Department of Computer Science and Technology, Nanjing University, China. He received his Ph.D. degree in computer science from Nanjing University in 2015. His research interests include social media mining, software repository mining, and software intelligence.

Hanghang Tong is currently an assistant professor at School of Computing, Informatics, and Decision Systems Engineering (CIDSE), Arizona State University since August 2014. Before that, he was an assistant professor at Computer Science Department, City College, City University of New York, a research staff member at IBM T.J. Watson Research Center and a Post-doctoral fellow in Carnegie Mellon University. He received his M.Sc and Ph.D. degree from Carnegie Mellon University in 2008 and 2009, both majored in Machine Learning. His research interest is in large scale data mining for graphs and multimedia. He has received several awards, including NSF CAREER award (2017), ICDM 2015 Highest-Impact Paper Award, four best paper awards (TUP14, CIKM12, SDM08, ICDM06), five bests of conference (KDD16, SDM15, ICDM15, SDM11 and ICDM10) and one best demo, honorable mention (SIGMOD17). He has published over 100 referred articles. He is an associated editor of SIGKDD Explorations (ACM), an action editor of Data Mining and Knowledge Discovery (Springer), and an associate editor of Neurocomping Journal (Elsevier); and has served as a program committee member in multiple data mining, databases and artificial intelligence venues (e.g., SIGKDD, SIGMOD, AAAI, WWW, CIKM, etc).



Feng Xu received the B.S. and M.S. degrees from Hohai University in 1997 and 2000, respectively. He received his Ph.D. degree from Nanjing University in 2003. He is a professor in the Department of Computer Science and Technology at Nanjing University. His research interests include trust management, trusted computing, and software reliability.



Jian Lu received his B.Sc., M.Sc. and Ph.D. degrees in Computer Science from Nanjing University, P.R. China. He is currently a professor in the Department of Computer Science and Technology and the Director of the State Key Laboratory for Novel Software Technology at Nanjing University. He serves on the Board of the International Institute for Software Technology of the United Nations University (UNU-IIST). He also serves as the director of the Software Engineering Technical Committee of the China Computer Federation. His research interests include software methodologies, software automation, software agents, and middleware systems.