Topical Word Trigger Model for Keyphrase Extraction

Zhiyuan Liu Chen Liang Maosong Sun
Department of Computer Science and Technology
State Key Lab on Intelligent Technology and Systems
National Lab for Information Science and Technology
Tsinghua University, Beijing 100084, China
{lzy.thu, chenliang.harry}@gmail.com, sms@tsinghua.edu.cn

ABSTRACT
Keyphrase extraction aims to find representative phrases for a document. Keyphrases are expected to cover main themes of a document. Meanwhile, keyphrases do not necessarily occur frequently in the document, which is known as the *vocabulary gap* between the words in a document and its keyphrases. In this paper, we propose Topical Word Trigger Model (TWTM) for keyphrase extraction. TWTM assumes the content and keyphrases of a document are talking about the same themes but written in different languages. Under the assumption, keyphrase extraction is modeled as a translation process from document content to keyphrases. Moreover, in order to better cover document themes, TWTM sets trigger probabilities to be topic-specific, and hence the trigger process can be influenced by the document themes. On one hand, TWTM uses latent topics to model document themes and takes the coverage of document themes into consideration; on the other hand, TWTM uses topic-specific word trigger to bridge the vocabulary gap between the words in document and keyphrases. Experiment results on real world dataset reveal that TWTM outperforms existing state-of-the-art methods under various evaluation metrics.

TITLE AND ABSTRACT IN CHINESE

采用基于主题的词触发模型进行关键词抽取

关键词抽取旨在发现文档中有代表性的词或者短语。抽取的关键词应当覆盖文档的主要话题。同时，关键词并不一定在文档中频繁出现，这就是所谓的文档与关键词间的“词汇鸿沟”问题。本文提出一种基于主题的词触发模型（TWTM）进行关键词抽取。该模型假设文档内容和关键词是在用不同的语言描述相同的话题。在这个假设下，关键词抽取就可以被建模为从文档到关键词的翻译过程。为了更好地覆盖文档话题，该模型的触发概率都是主题相关的，从而使触发过程受到文档主题的影响。一方面，该模型利用隐含主题对文档话题进行建模，从而将文档主题的覆盖度考虑在内；另一方面，该模型采用主题相关的词触发建立起了文档与关键词的桥梁。在真实数据上的实验结果表明，该模型优于已有关键词抽取方法。

KEYWORDS: keyphrase extraction, latent topic model, word trigger model.

KEYWORDS IN CHINESE: 关键词抽取, 隐含主题模型, 词触发模型.

1 Introduction

For information retrieval and management, people usually annotate a collection of keyphrases to a document as its brief summary. Keyphrases can be found in most digital libraries and information retrieval systems (Turney, 2000; Nguyen and Kan, 2007). Web information, most of which is in the form of text, is growing at a rapid rate. For the large volume of documents, it will be inefficient for human editors to manually index keyphrases. Therefore, automatically extracting keyphrases for documents is proposed as a challenging task in natural language processing. The task is also referred to as *keyphrase extraction*.

When we enter Web 2.0 era, social tagging is invented to help users manage and share information. The social tags can be regarded as a type of keyphrases. Various methods have been proposed for automatic social tag suggestion, which can be regarded as a special type of keyphrase extraction. In social tag suggestion, given a document, the system will select keyphrases from a controlled tag list instead of document itself. It indicates that keyphrases do not necessarily occur in the given document. It is obvious that it provides a more flexible and convenient scheme compared to traditional keyphrase extraction, and thus becomes the main application of keyphrase extraction. In this paper, we will focus on the new setting of keyphrase extraction, which is named as *keyphrase extraction from a controlled vocabulary*. In the following paper, unless specifically noted, we use keyphrase extraction as referred to the new setting.

As a summary of document, keyphrases are expected to represent and cover the main themes of the given document. Suppose there is an article talking about the "Apple" company and its smartphone "iPhone". The extracted keyphrases are expected to cover the both themes, i.e., "Apple" and "iPhone". This indicates that a set of keyphrases that focuses on only one theme will not be adequate. Meanwhile, representative keyphrases do not necessarily occur frequently in the document. Take the article for example again, it may mention "iPhone" (smartphone of Apple), "iPad" (tablet computer of Apple) and "Steve Jobs" (founder of Apple) for many times, but refer to "Apple" not so frequently. Nevertheless, it is intuitive that "Apple" should be a representative keyphrase of the document. We refer the phenomenon as a *vocabulary gap* between words in document and keyphrases. In summary, given a document, keyphrase extraction should: (1) find a set of representative keyphrases that can better cover the main themes of the document. (2) The selection of keyphrases should primarily rely on their semantic relatedness with the document rather than being constrained by their occurrence frequencies in the document. This requires keyphrase extraction can bridge the vocabulary gap between document content and keyphrases.

Many unsupervised methods have also been extensively explored for keyphrase extraction. The most simple unsupervised method is ranking the candidate keyphrases according to TFIDF (Salton and Buckley, 1988) and then selecting top-ranked ones as keyphrases. There are also graph-based methods (Mihalcea and Tarau, 2004; Wan and Xiao, 2008b,a; Liu et al., 2010), clustering-based methods (Grineva et al., 2009; Liu et al., 2009) and latent topic models (Heinrich, 2005; Blei and Lafferty, 2009) proposed for keyphrase extraction. Most of these methods take frequencies of candidate keyphrases as the crucial decision criteria, and thus tend to select those high-frequency ones as keyphrases. Given sufficient annotation data for training, we can adopt the classification-based approach for keyphrase extraction. For example, some methods (Frank et al., 1999; Witten et al., 1999; Turney, 2000) regard keyphrase extraction as a binary classification problem (is-keyphrase

or non-keyphrase). Keyphrase extraction can also be considered as a multi-label classification problem (Tsoumakas and Katakis, 2007), in which each keyphrase is regarded as a category label. Various methods such as Naive Bayes (Garg and Weber, 2008) and *k*NN (Li et al., 2009) have been explored. Some researchers proposed using latent topics to build semantic relations between words and tags. The representative methods include TagLDA (Krestel et al., 2009; Si and Sun, 2009) and Content Relevance Model (CRM) (Iwata et al., 2009). However, these methods usually suffer from the over-generalization problem.

Recently, a new approach based on word alignment models (WAM) in statistical machine translation (SMT) has been proposed for keyphrase extraction (Ravi et al., 2010; Liu et al., 2011b,a, 2012). WAM-based methods assume the content and keyphrases of a document are describing the same themes but written in different languages. Under this assumption, WAM-based methods regard keyphrase extraction as a translation process from document content to keyphrases. This process is modeled as a trigger from important words in document content to keyphrases according to trigger probabilities between words and keyphrases. WAM-based methods will learn trigger probabilities from sufficient document-keyphrase pairs. With the trigger probabilities, given a novel document, WAM-based methods are able to extract relevant keyphrases that do not necessarily occur so frequently in the document.

Although achieving significant improvement in bridging the vocabulary gap between document and keyphrases, WAM-based methods, however, cannot well guarantee the coverage of document themes. The crucial reason is analyzed as follows. WAM-based methods, formalizing trigger probabilities at *word level*, consider each single word in document and project from document content to keyphrases. However, the coverage of document themes should be appreciated at *topic level*, which is beyond the power of WAM-based methods.

A promising approach for representing document themes is latent topic models (Blei et al., 2003). In topic models, both words/keyphrases and documents are represented as probabilistic distributions over latent topics. Topic models are widely adopted as a guaranteed approach to represent document themes. Topic models themselves can also be used for keyphrase extraction, referred to as topic-based methods, by simply ranking keyphrases according to their semantic relevance with the document themes in terms of latent topics. Topic-based methods can be regarded as a trigger process at topic level, contrast to the trigger process at word level in WAM-based methods. Since common keyphrases receive larger probabilities given a topic, topic-based methods tend to select those keyphrases that are too general to tightly capture the main themes of the document. For example, it may select "IT" as keyphrase for the above mentioned document, which is so general that cannot reflect the document themes well.

Is there a way to leverage the power of both word-level projection and topic-level coverage in keyphrase extraction? Can the two techniques, i.e., word alignment models and latent topic models, be integrated together to complement each other for keyphrase extraction? To address the problems, we propose Topical Word Trigger Model (TWTM) for keyphrase extraction. TWTM inherits the advantage of WAM-based methods, and also incorporates latent topic models so as to promise the coverage of document themes.

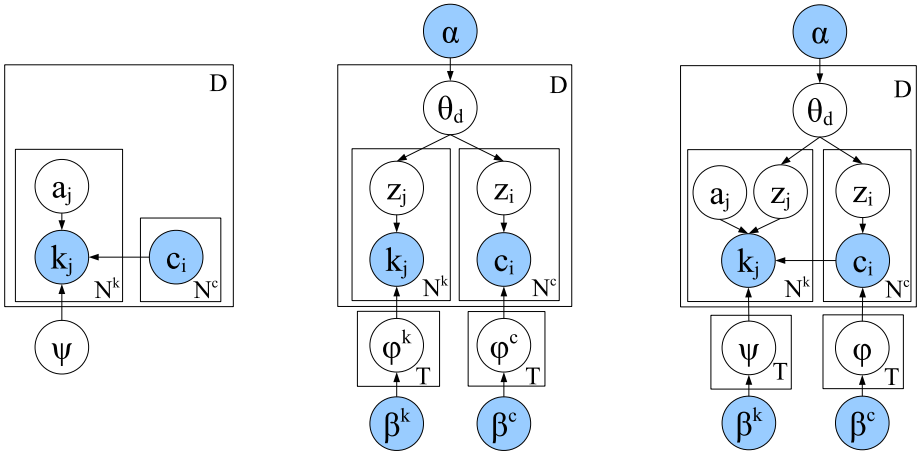
To compare and analyze the characteristics of different approaches for keyphrase extraction, we also introduce the method based on word alignment models, i.e., Word Trigger Model (WTM), and the method based on polylingual topic models, i.e., Topic Trigger Model (TTM).

As these names suggest, WTM identifies keyphrases by triggering at word level; while TTM triggering at topic level. TWTM, integrating their advantages, performs both word-level and topic-level triggers to extract keyphrases.

To demonstrate the effectiveness of our method, we carry out experiments on a real-world dataset crawled from a website with keyphrases having been annotated collaboratively by users. Experiment results show that TWTM can identify appropriate keyphrases with better coverage of document themes compared to existing WAM-based methods.

2 Our Method

In this section we first introduce two simple trigger methods, WTM and TTM, in which WTM performs triggering at word level while TTM at topic level. Afterwards, we introduce our method TWTM.



(a) WTM (b) TTM (c) TWTM

Figure 1: Trigger Models for keyphrase extraction.

2.1 Notations and Definitions

Before introducing baselines and our method, we give some notations. We denote a document as $d \in D$, where D is the document set. For each document d, we denote its content as a sequence of words $\mathbf{c} = \{c_i\}_{i=1}^{N^c}$ and its keyphrases as a set $\mathbf{k} = \{k_i\}_{i=1}^{N^k}$. The vocabulary of words in documents is denoted as W, and the vocabulary of keyphrases as V. Each word c_i in documents is an instance of a word type w in W, i.e., $c_i = w \in W$; each keyphrase k_i of documents is an instance of a keyphrase type v in V, i.e., $k_i = v \in V$.

We define keyphrase extraction as follows. Given a document d with its content $\mathbf{c}$, keyphrase extraction aims to seek a set of keyphrases $\mathbf{k}$ that maximizes $\Pr(\mathbf{k}|\mathbf{c})$. By simply assuming the keyphrases are independent conditional over d, we have $\Pr(\mathbf{k}|\mathbf{c}) = \prod_{k\in\mathbf{k}} \Pr(k|\mathbf{c})$. The optimal set of keyphrases $\mathbf{k}^*$ can be represented as follows:

$$\mathbf{k}^* = \arg\max_{\mathbf{k}} \Pr(\mathbf{k}|\mathbf{c}) = \arg\max_{\mathbf{k}} \prod_{k\in\mathbf{k}} \Pr(k|\mathbf{c}). \quad (1)$$

Suppose the number of keyphrases is pre-defined as N^k, we can simply find $\mathbf{k}^*$ by ranking

each candidate keyphrase $v \in V$ according to its score $\Pr(v|\mathbf{c})$ in descending order and selecting top-N^k keyphrases.

2.2 Word Trigger Model

Word Trigger Model (WTM) is inspired by IBM Model-1 (Brown et al., 1993), the most widely used word alignment models in SMT. WTM assumes the content and the keyphrases of a document are describing the same themes while written in two different languages: document content in one language while keyphrases in the other. From this perspective, keyphrase extraction can be regarded as a translation process from a given document content to keyphrases.

In more detail, the translation process is modeled as a trigger process as follows. First, WTM finds several important words in the document content as *trigger words*. Then, activated by these trigger words, WTM maps the document content into keyphrases. A *trigger* in the translation process can be regarded as a mapping function from the words in document to keyphrases.

WTM formalizes a trigger as a hidden variable $\mathbf{a}$. By assuming each keyphrase of a document is triggered by only one word in the content, a $\mathbf{a}$ maps each keyphrase at position j (i.e., k_j) as triggered by a word at position i in document content c (i.e., c_i), denoted as $a_j = i$. Given a document, its content $\mathbf{c}$ and keyphrases $\mathbf{k}$ can be connected by a trigger variable $\mathbf{a}$. The probability of triggering $\mathbf{k}$ from $\mathbf{c}$ can be formalized as

$$\Pr(\mathbf{k}|\mathbf{c}) = \sum_{\mathbf{a}} \Pr(\mathbf{k}, \mathbf{a}|\mathbf{c}) = \sum_{\mathbf{a}} \Pr(\mathbf{k}|\mathbf{a}, \mathbf{c}) \Pr(\mathbf{a}|\mathbf{c}). \tag{2}$$

Following the same assumptions of IBM Model-1 (Brown et al., 1993), for each document $d \in D$, WTM assumes the content $\mathbf{c}$ of d already exists, and the keyphrases $\mathbf{k}$ are generated from $\mathbf{c}$ as follows:

1. For each document d with content $\mathbf{c}$ and N^k keyphrases:
   (a) For j from 1 to N^k:
      i. Sample each word trigger link a_j from $1, \ldots, N^c$ according to a uniform distribution.
      ii. Sample each $k_j = v$ according to trigger probability $\Pr(k_j = v|c_{a_j} = w, \psi)$.

Here we denote the trigger probability from a word $w \in W$ to a keyphrase $v \in V$ as $\psi_{vw} = \Pr(v|w)$, and the trigger probabilities from W to V form a matrix ψ. The corresponding graphical representation is shown in Figure 1a. The boxes are "plates" representing replicates. In this graphical model, the variables $\mathbf{k}$ and $\mathbf{c}$ are shaded indicating that they are observed; while the unshaded variables, including $\mathbf{a}$ and ψ, are latent (i.e., unobserved). Under the same assumptions of IBM Model-1, we write

$$\Pr(\mathbf{k}|\mathbf{c}) \propto \frac{1}{(N^c)^{(N^k)}} \prod_{j=1}^{N^k} \sum_{i=1}^{N^c} \Pr(k_j = v|c_i = w) = \frac{1}{(N^c)^{(N^k)}} \prod_{j=1}^{N^k} \sum_{i=1}^{N^c} \psi_{vw}. \tag{3}$$

We see that, in WTM, trigger probabilities $\psi_{vw} = \Pr(k_j = v|c_i = w)$ are the key parameters for learning. WTM has global optimum, and is efficient and easily scalable to large training

data. We use Expectation-Maximization (EM) (Dempster et al., 1977) to estimate trigger probabilities ψ.

Using the estimated ψ, when given a novel document d with its content $\mathbf{c}$, we can rank each candidate keyphrase v as follows:

$$\Pr(v|\mathbf{c}) = \sum_{w \in \mathbf{c}} \Pr(v|w, \psi) \Pr(w|\mathbf{c}) = \sum_{w \in \mathbf{c}} \psi_{vw} \Pr(w|\mathbf{c}), \tag{4}$$

where $\Pr(w|\mathbf{c})$ indicates the weight of the word w in $\mathbf{c}$, which can be calculated using the TFIDF score of w in $\mathbf{c}$. From the ranking list in descending order, we can select the top-ranked ones as keyphrases of the given document.

2.3 Topic Trigger Model

WTM triggers keyphrases at the word level. We can also trigger at the topic level, and thus propose Topic Trigger Model (TTM) for keyphrase extraction. TTM is inspired by Polylingual Topic Models (Mimno et al., 2009), which is originally proposed to model parallel documents in multiple languages. TTM is an extension of latent Dirichlet allocation (LDA) (Blei et al., 2003).

Suppose there are T latent topics in TTM, and the number of topics $|T|$ can be pre-defined by users. TTM assumes that the content $\mathbf{c}$ and keyphrases $\mathbf{k}$ of a document d share the same distribution over $|T|$ topics (i.e., θ_d), which is drawn from a symmetric Dirichlet prior with concentration parameter α. TTM also assumes that each topic $t \in T$ corresponds to two different multinomial distributions over words, one for keyphrases (i.e., ϕ_t^k) and another for content (i.e., ϕ_t^c), each of which is drawn from a specific symmetric Dirichlet with concentration parameter, β^k or β^c. TTM can be viewed as a generative process of both document content and keyphrases as follows:

1. Sample word distribution ϕ_t^c from $Dirichlet(\beta^c)$ and sample keyphrase distribution ϕ_t^k from $Dirichlet(\beta^k)$ for each topic $t \in T$.
2. For each document $d \in D$ with N^c words and N^k keyphrases:
   (a) Sample topic distribution θ_d from $Dirichlet(\alpha)$.
   (b) For i from 1 to N^c
      i. Sample a topic $z_i = t$ from $Multinomial(\theta_d)$.
      ii. Sample a word $c_i = w$ according to multinomial distribution $\Pr(c_i = w|z_i = t, \phi^c)$.
   (c) For j from 1 to N^k
      i. Sample a topic $z_j = t$ from $Multinomial(\theta_d)$.
      ii. Sample a keyphrase $k_j = v$ according to multinomial distribution $\Pr(k_j = v|z_j = t, \phi^k)$.

The corresponding graphical model is shown in Figure 1b, where the observed variables (i.e., words $\mathbf{c}$, keyphrases $\mathbf{k}$ and hyper-parameters β^k and β^c) are shaded.

Given the observed words in a collection of documents, the task of TTM learning is to compute the posterior distribution of the latent topic assignments $\mathbf{z}$, the topic mixtures θ_d of

each document d, and the distributions over words ϕ_t^c and ϕ_t^k of each topic t. By assuming a Dirichlet prior β on ϕ, ϕ can be integrated according to the Dirichlet-multinomial conjugacy. In this paper, we use Gibbs Sampling to estimate parameters, which has been widely used as an inference method for many latent topic models. In Gibbs sampling, it is usual to integrate out the mixtures θ and topics ϕ and just sample the latent variables $\mathbf{z}$. The process is thus called *collapsed*.

Gibbs Sampling iteratively performs latent topic assignments for each word in the document set, and estimates the distributions over words of each topic (i.e., $\phi_{wt}^c = \Pr(c_i = w|z_i = t)$ and $\phi_{vt}^k = \Pr(k_j = v|z_j = t)$), and the distribution over topics of each document (i.e., $\theta_{td} = \Pr(z_i = t|d)$). Take a word token $c_i = w$ in d for example, given the current state of all but the variable z_i, the conditional probability of $z_i = t$ is

$$\Pr(z_i = t|c_i = w, \mathbf{c}^{\neg i}, \mathbf{z}^{\neg i}, \mathbf{k}) \propto \frac{N_{wt}^{c,\neg i} + \beta^c}{N_t^c - 1 + |W|\beta^c} \times \frac{N_{td}^{\neg i} + \alpha}{N_d^c + N_d^k - 1 + |T|\alpha}, \tag{5}$$

where z is the current topic assignments for all tokens in the document set; N_{wt}^c is the number of occurrences of word w that are assigned with topic t; N_t^c is the number of occurrences of all words that are assigned with topic t; N_{td} is the number of occurrences of topic t assigned in the current document d; N_d^c and N_d^k are the numbers of all tokens in the content and keyphrases of d, respectively; $\neg i$ indicates taking no account of the current position i.

According to the posterior probability $\Pr(z_i = t|c_i = w, \mathbf{c}^{\neg i}, \mathbf{z}^{\neg i}, \mathbf{k})$, we re-sample the topic assignment z_i of the c_i in d. Whenever z_i of c_i is assigned with a new topic drawn from Equation (5), N_{wt}^c and N_{td} are updated. We perform topic assignments in the same way for each word k_i in k of d. After enough sampling iterations to burn in the Markov chain, ϕ^c, ϕ^k and θ are estimated as follows:

$$\phi_{wt}^c = \frac{N_{wt}^c + \beta^c}{N_t^c + |W|\beta^c}, \quad \phi_{vt}^k = \frac{N_{vt}^k + \beta^k}{N_t^k + |V|\beta^k}, \quad \theta_{td} = \frac{N_{td} + \alpha}{N_d^c + N_d^k + |T|\alpha}. \tag{6}$$

When finishing the learning process, we obtain the distributions over words of each topic, i.e., ϕ^k and ϕ^c. Suppose we are asked to extract keyphrases from a novel document with only content $\mathbf{c}$. First, we infer topic assignments for each word in $\mathbf{c}$ with Gibbs Sampling. With the topic assignments, we summarize the distribution over topics of the content $\mathbf{c}$ as

$$\Pr(t|\mathbf{c}) = \frac{N_{td}^c + \alpha}{N_d^c + |T|\alpha}. \tag{7}$$

Triggered by the topics of $\mathbf{c}$, we rank each candidate keyphrase $v \in V$ as follows:

$$\Pr(v|\mathbf{c}) = \sum_{t \in T} \Pr(v|t, \phi^k) \Pr(t|c) = \sum_{t \in T} \phi_{vt}^k \theta_{td}, \tag{8}$$

and then select the top-ranked as keyphrases of the given document.

2.4 Topical Word Trigger Model

WTM and TTM perform trigger operations at either word or topic level. WTM addresses the problem of vocabulary gap between documents and keyphrases, and can thus suggest

keyphrases that are uncommon or even not showing up in the given document. TTM, on the other hand, takes the main themes of the given document in consideration when extracting keyphrases. In order to aggregate the advantages of the both methods, extended from WTM and TTM, we propose Topical Word Trigger Model (TWTM) for keyphrase extraction.

Similar to TTM, TWTM also assumes that topics are sampled at the word level. Each document is represented as a multinomial distribution over T latent topics. On the document content side, each topic $t \in T$ corresponds to a multinomial distribution over words, which is similar to TTM. On the keyphrase side, each topic $t \in T$ corresponds to a topic-specific translation table ψ_t. Given each document $d \in D$, the generative process of both document content and keyphrases is as follows:

1. Sample word distribution ϕ_t^c from $Dirichlet(\beta^c)$ for each topic $t \in T$.
2. Sample keyphrase distribution ψ_w^t from $Dirichlet(\beta^k)$ for each topic $t \in T$ and each word $w \in W$.
3. For each document $d \in D$ with N^c words and N^k keyphrases:
   (a) Sample topic distribution θ_d from $Dirichlet(\alpha)$.
   (b) For i from 1 to N^c
      i. Sample a topic $z_i = t$ from $Multinomial(\theta_d)$.
      ii. Sample a word $c_i = w$ according to multinomial distribution $\Pr(c_i = w | z_i = t, \phi^c)$.
   (c) For j from 1 to N^k
      i. Sample a topic $z_j = t$ from $Multinomial(\theta_d)$.
      ii. Sample each word trigger link a_j from all words in d that are generated from t according to a uniform distribution.
      iii. Sample each $k_j = v$ according to trigger probability $\Pr(k_j = v | c_{a_j} = w, \psi^t)$.

The graphical model is shown in Figure 1c. Topic-dependent translation probabilities ψ_t are the key parameters. Each ψ_t maintains the translation probability from each word $c_i = w$ in contents to each keyphrase $k_j = v$ under topic t, i.e., $\psi_{vw}^t = \Pr(k_j = v | c_i = w, t)$.

Given the observed words and keyphrases in a collection of documents, the task of TWTM learning is to compute the posterior distribution of the latent topic assignments $\mathbf{z}$, the topic mixtures θ_d of each document d, the distribution over words ϕ_t of each topic t, and the trigger probabilities ψ^t of each topic t. We can also use Gibbs Sampling to estimate parameters in TWTM. On the document content side, we can perform topic assignments $\mathbf{z}$ for each word as in TTM using Equation (5). On the keyphrase side, the problem is more complicated. Suppose we will assign each keyphrase k_j with a topic z_j and a trigger a_j. Take a keyphrase $k_j = v$ for example, given the current state of all but the variable z_j and a_j, the conditional probability of $z_j = t, a_j = i$ is calculated as follows,

$$\Pr(z_j = t, a_j = i | k_j = v, \mathbf{k}^{\neg j}, \mathbf{z}^{\neg j}, \mathbf{a}^{\neg j}, c_i = w, \mathbf{c}) \propto \frac{N_{vw}^{t,\neg j} + \beta^k}{N_w^t - 1 + |V|\beta^k} \times \frac{N_{td}^{\neg j} + \alpha}{N_d - 1 + |T|\alpha}, \tag{9}$$

where z is the current topic assignments for all translation pairs in the document set; N_{vw}^t is the number of occurrences that w is translated to v given topic t; N_w^t is the number of occurrences of word w in all translation pairs given topic t; N_{td} is the number of occurrences of topic t assigned in the current document d; N_d is the number of all translation pairs in

the current document d; $\neg j$ also indicates taking no account of the current position j. Given the conditional probability of $z_j = t, a_j = i$, we formalize the marginal probability of $z_j = t$ as follows,

$$\Pr(z_j = t | k_j = v, k^{\neg j}, z^{\neg j}, a^{\neg j}, c) \propto \sum_{i=1}^{N^c} \frac{N_{vc_i}^{t,\neg j} + \beta^k}{N_{c_i}^t - 1 + |V|\beta} \times \frac{N_{td}^{\neg j} + \alpha}{N_d - 1 + |T|\alpha}. \quad (10)$$

After re-assigning the topic $z_j = t$ for the current keyphrase according to Equation (10), we can further compute the trigger probability as follows:

$$\Pr(a_j = i | z_j = t, k_j = v, k^{\neg j}, z^{\neg j}, a^{\neg j}, c) = \frac{N_{vc_i}^{t,\neg j} + \beta}{N_{c_i}^t - 1 + |V|\beta^k}. \quad (11)$$

According to Equation (11), we re-assign trigger word c_i for the current keyphrase k_j. After enough sampling iterations to burn in the Markov chain, ϕ^c, ψ^t and θ are estimated as follows:

$$\phi_{wt}^c = \frac{N_{wt}^c + \beta^c}{N_t^c + |W|\beta^c}, \quad \psi_{vw}^t = \frac{N_{vw}^t + \beta}{N_w^t + |V|\beta}, \quad \theta_{td} = \frac{N_{td} + \alpha}{N_d^k + N_d^c + |T|\alpha}. \quad (12)$$

The potential size of topical trigger probabilities ψ is $|V| \times |W| \times |T|$. The size is comparative larger than ψ in WTM, and thus faces more serious problem of data sparsity. To remedy the problem, we use interpolation smoothing technique for ψ of TWTM. In this paper, we employ smoothing using ψ of WTM as follows:

$$\Pr_{SMOOTH}(v|w,t) = \lambda \Pr_{TWTM}(v|w,t) + (1-\lambda) \Pr_{WTM}(v|w,t), \quad (13)$$

where $\Pr_{SMOOTH}(v|w,t)$ is the smoothed topical trigger probabilities, $\Pr_{TWTM}(v|w,t)$ is the original topical trigger probabilities of TWTM, $\Pr_{WTM}(v|w,t)$ is the trigger probabilities of WTM. λ is the smoothing factor ranging from 0.0 to 1.0. When $\lambda = 0.0$, $\Pr_{SMOOTH}^t(v|w)$ will be reduced to non-topic trigger probabilities; and when $\lambda = 1.0$, there will be no smoothing in $\Pr_{SMOOTH}(v|w,t)$.

In TWTM, we perform keyphrase extraction as follows. Suppose we need perform keyphrase extraction for a document d with its content $\mathbf{c}$. We perform Gibbs Sampling to iteratively estimate the topic distribution of d (i.e., θ_d) according to document content $\mathbf{c}$. Afterwards, we select N^k keyphrases using the ranking score of each keyphrase v:

$$\Pr(v|\mathbf{c}) = \sum_{w \in \mathbf{c}} \sum_{t \in T} \Pr(v|w,t) \Pr(w|\mathbf{c}) \Pr(t|\mathbf{c}) = \sum_{w \in \mathbf{c}} \sum_{t \in T} \psi_{vw}^t \theta_{td} \Pr(w|\mathbf{c}), \quad (14)$$

where $\Pr(w|\mathbf{c})$ is the weight of the word w in document content c, which can be estimated by the TFIDF score of w in $\mathbf{c}$; $\Pr(t|\theta_d)$ is the probability of the topic t given the document d.

3 Experiments and Analysis

3.1 Dataset and Experiment Setting

To evaluate the performance of TWTM for keyphrase extraction, we carry out experiments on a real world dataset, crawled from `douban.com`, the largest product review website in China. Each product contains a description which is considered as a document content, and also contains a set of keyphrases annotated by users collaboratively which are considered

as standard keyphrases. The dataset consists of annotations for three types of products, i.e., book, movie and music. The statistics of the dataset is shown in Table 1, where $|D|$, $|W|$, $|V|$, $\hat{N}^c$ and $\hat{N}^k$ are the number of documents, the vocabulary of contents, the vocabulary of keyphrases, the average number of words and keyphrases in each document, respectively. In Table 1, we use DOUBAN to represent the whole dataset and use BOOK, MOVIE and MUSIC to show the statistics for instances for different product types.

Data	$\|D\|$	$\|W\|$	$\|V\|$	$\hat{N}^c$	$\hat{N}^k$
DOUBAN	71,525	160,276	99,457	86.30	10.53
BOOK	26,807	81,846	41,199	83.13	8.95
MOVIE	18,933	86,339	37,034	86.04	16.03
MUSIC	25,785	106,523	31,228	89.77	8.13

Table 1: Statistical information of dataset.

To evaluate the performance of our method and compare with other methods, we randomly select 1,000 instances from each of three types of products to form the test set with 3,000 instances, and use the rest of the dataset as training set.

In our experiments we select three evaluation metrics. The first metric is precision, recall and F-measure represented as $p = c_{correct}/c_{extract}$, $r = c_{correct}/c_{standard}$ and $F = 2pr/(p+r)$, where $c_{correct}$ is the total number of keyphrases that are correctly suggested by a method, $c_{extract}$ is the total number of automatic extracted keyphrases, and $c_{standard}$ is the total number of human-labeled standard keyphrases.

In fact, ranking order of extracted keyphrases also indicates the performance of different methods. A method is regarded better than another one if it ranks correct keyphrases higher. However, precision/recall/F-measure does not take the order of extracted keyphrases into account. To address the problem, we select the following two additional metrics. One metric is *binary preference measure* (Bpref) (Buckley and Voorhees, 2004). Bpref can consider the order of extracted keyphrases for evaluation. For a document, if there are R correct keyphrases within M extracted keyphrases by a method, in which r is a correct keyphrase and n is an incorrect keyphrase. It is defined as $\text{Bpref} = \frac{1}{R}\sum_{r\in R}\left(1 - \frac{|n \text{ ranked higher than } r|}{M}\right)$.

The other metric is *mean reciprocal rank* (MRR) (Voorhees, 2000) which is usually used to evaluate how the first correct keyphrase for each document is ranked. For a document d, $rank_d$ is denoted as the rank of the first correct keyphrase with all extracted keyphrases, It is defined as $\text{MRR} = \frac{1}{|D|}\sum_{d\in D}\frac{1}{rank_d}$, where D is the document set for keyphrase extraction.

3.2 Case Studies

Before quantitative evaluation, we perform case studies by looking into the topics learned by TWTM. By setting $T = 100$ of TWTM, We select two topics, i.e., Topic-59 and Topic-92 for study. In first several rows of Table 2, we list the top-10 words and top-10 keyphrases given the two topics separately (i.e., ranked by $\Pr(w|t)$ and $\Pr(v|t)$). From the top words and keyphrases, we can conclude that Topic-59 is about "art design" and Topic-92 is about "computer programming".

What will the topics influence the trigger probabilities? We pick a word "graphics" for example. In the context of the topic "art design", the word "graphics" always correlates with "design", "color" and "art"; while in the context of the topic "computer programming", the

word "graphics" generally refers to "computer graphics" and thus correlates to "programming", "software" and "programming language". At the bottom of Table 2, we show the top-6 keyphrases triggered by the word "graphics" with respect to the two topics. The value in the bracket after each keyphrase v is the probability $\psi^t_{vw} = \Pr(v|w,t)$, which is the topical specific trigger probability from w (here is the word "graphics") to v under the topic t. From the top triggered keyphrases by "graphics" under two topics, we can see they are discriminative with each other, and have intense topic characteristics.

	Topic-59		Topic-92	
Top	Words	Keyphrases	Words	Keyphrases
1	design	design	program	computer
2	creativity	creativity	develop	program
3	designer	designing	application	software engineer
4	magazine	handcraft	object	C++
5	fashion	graphic design	technology	programming
6	game	fashion	design	program design
7	work	game	system	program develop
8	color	product design	function	Linux
9	vision	industrial design	software	computer science
10	advertise	magazine	method	Alan
graphics	design (0.482)		game programming (0.201)	
	color science (0.089)		programming language (0.107)	
	font design (0.084)		Web 2.0 (0.094)	
	product design (0.077)		C (0.078)	
	landscape design (0.050)		Linux (0.077)	
	art design (0.039)		Computer Graphics (0.049)	

Table 2: Examples of topics learned with TWTM.

After investigating the topics, we look into keyphrase extraction results given a product description. Here we select a Japanese classical literature *The Tale of Genji* for example, which was written by *Murasaki Shikibu* in the early years of the 11th century [1]. The book recounts the life and love stories of a son of the Japanese emperor. In Table 3, we show the top-10 keyphrases extracted by WTM, TTM and TWTM, in which we use (−) to highlight the inappropriate keyphrases.

Method	Extracted Keyphrases
WTM	The Tale of Genji, classic, Japan, foreign literature, Murasaki Shikibu, politics (-), love, political science (-), eason (-), political philosophy (-)
TTM	novel, Japan, foreign literature, history, love, sociology (-), culture, literature, Russia (-), female (-)
TWTM	novel, foreign literature, The Tale of Genji, history, Japan, classic, literature, love, Murasaki Shikibu, politics (-)

Table 3: Examples of extracted keyphrases for the book *The Tale of Genji*.

From Table 3 we observe that: (1) WTM can suggest keyphrases that are closely related to the book, such as "The Tale of Genji" and "Murasaki Shikibu". However, due to not considering document themes, WTM will extract irrelevant keyphrases such as "politics", "political

[1] `http://en.wikipedia.org/wiki/The_Tale_of_Genji`.

science", "eason" and "political philosophy". (2) TTM triggers keyphrases with the favor of latent topics, which are usually too general to commendably represent the document main themes. We can see TTM fail to extract specific keyphrases such as "The Tale of Genji" and "Murasaki Shikibu". What is worse, TTM over-generalizes the document themes and extract inappropriate keyphrases "sociology", "'Russia" and "female". (3) Taking advantages of both WTM and TTM, TWTM can extract specific and representative keyphrases and at the same time guarantee the coverage of document themes. We can see that TWTM achieves a smart balance between word-level projections and topic-level coverage.

3.3 Parameter Influences

There are two crucial parameters in TWTM, the number of topics T and the smoothing factor λ. In Table 4 and Table 5, we demonstrate the performance of TWTM for keyphrase extraction when parameters change.

T	Precision	Recall	F-measure	Bpref	MRR
10	0.313	0.304	0.309	0.313	0.825
30	0.337	0.325	0.331	0.337	0.837
50	0.339	0.329	0.334	0.339	0.827
70	0.351	0.339	0.345	0.351	**0.840**
100	**0.354**	**0.343**	**0.349**	**0.354**	0.838

Table 4: The influence of topic number T of TWTM for keyphrase extraction when $N^k = 10$ and smoothing factor $\lambda = 0.4$.

λ	Precision	Recall	F-measure	Bpref	MRR
0.0	0.310	0.254	0.279	0.310	0.676
0.2	0.318	0.314	0.316	0.318	0.823
0.4	0.354	0.343	0.349	0.354	**0.838**
0.6	**0.364**	**0.349**	**0.357**	**0.364**	0.812
0.8	0.350	0.334	0.342	0.351	0.764
1.0	0.323	0.306	0.314	0.324	0.731

Table 5: The influence of smooth factor λ of TWTM for keyphrase extraction when $N^k = 10$ and the number of topics $T = 100$.

From Table 4, we can see that, as the number of topics T increases from $T = 10$ to $T = 100$, the performance roughly improves. This indicates that the granularity of topics will influence the keyphrase extraction performance. When $T = 70$ and $T = 100$, the performance achieves a relatively stable good status. Hence, when comparing TWTM with other methods, we set $T = 100$ for TWTM.

As shown in Table 5, when the smoothing factor is set with $\lambda = 0.4$ or $\lambda = 0.6$, TWTM achieves the relatively best performance. When either $\lambda = 0.0$ (i.e., WTM), or $\lambda = 1.0$ (i.e., non-smoothed TWTM), the performance is much poorer compared to smoothed TWTM. This reveals that it is necessary to address the sparsity problem of TWTM by smoothing with WTM. Therefore, when comparing TWTM with other methods, we set $\lambda = 0.4$ for TWTM.

3.4 Performance Comparison

Besides WTM and TTM, we also select three representative methods as baselines for comparison: the classification-based method Naive Bayes (NB) (Garg and Weber, 2008), the topic-based method CRM (Iwata et al., 2009) and the word-projection method TAM (Si et al., 2010). We set $T = 1,024$ for CRM which achieves its best performance.

In Figure 2 we show the precision-recall curves of NB, CRM, TAM, WTM, TTM and TWTM on the dataset. Each point of a precision-recall curve represents extracting different number of keyphrases ranging from $N^k = 1$ (bottom right, with higher precision and lower recall) to $N^k = 10$ (upper left, with higher recall but lower precision), respectively. The closer the curve to the upper right, the better the overall performance of the method.

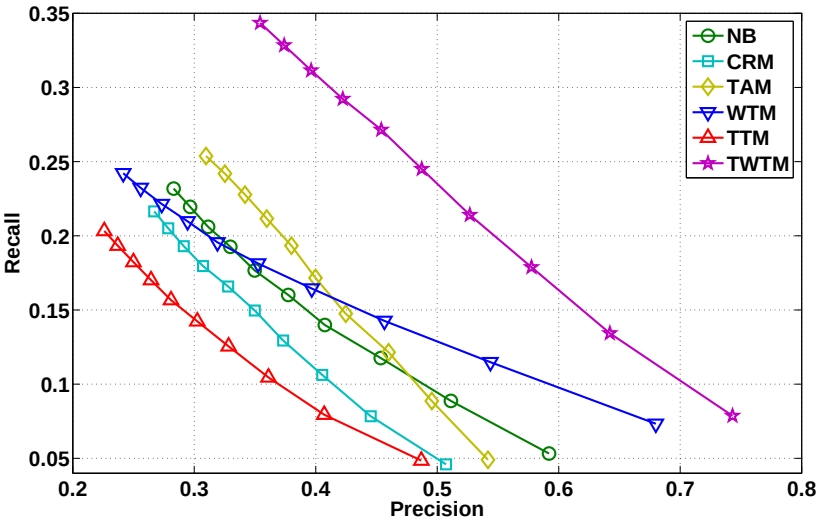


Figure 2: Precision-recall curves for keyphrase extraction.

Figure 2 clearly shows that TWTM outperforms other methods significantly. The interesting phenomena is that, when N^k is getting larger, the advantages of TWTM are more obvious compared to baselines. We know that when a system is asked to extract more keyphrases (i.e., N^k is larger), it is becoming important for extracted keyphrase to have a good coverage of document themes. Otherwise, the performance will drop sharply. This is the issue suffered by WTM. We can see that, although WTM is relatively excellent when suggesting top several keyphrases, it performs poor when suggesting more keyphrases due to the poor ability on ensuring coverage.

In Table 6 we list the comparison results of various methods when extracting $N^k = 10$ keyphrases. We can observe that TWTM outperforms the best baseline TAM by 7% of F-measure. Moreover, as mentioned above, the dataset consists of three types of products. In Table 6 we also demonstrate the results of TWTM on the test instances divided by product types, denoted as "BOOK", "MOVIE" and "MUSIC". The performance is consistently decent on the three types of instances. We also observe that F-measure scores on the three types of instances are proportional to the size of their training instances as shown in Table 1. Apparently, more training instances will enhance sufficiently learning of TWTM, which may

Method	Precision	Recall	F-measure	Bpref	MRR
NB	0.283	0.232	0.255	0.283	0.702
CRM	0.267	0.216	0.239	0.267	0.648
TAM	0.310	0.254	0.279	0.310	0.676
WTM	0.242	0.242	0.242	0.242	0.785
TTM	0.226	0.203	0.214	0.226	0.638
TWTM	**0.354**	**0.343**	**0.349**	**0.354**	**0.838**
BOOK	0.365	0.428	0.394	0.365	0.861
MOVIE	0.356	0.274	0.310	0.356	0.820
MUSIC	0.341	0.326	0.334	0.341	0.831

Table 6: Comparison results when extracting $N^k = 10$ keyphrases.

also eventually improve the performance of keyphrase extraction.

Conclusion and Future Work

This paper focuses on keyphrase extraction from a controlled vocabulary. The proposed TWTM has two features: (1) TWTM uses latent topics to represent document themes, and thus takes the coverage of document themes into consideration; (2) TWTM models topic-specific word triggers, which are more discriminative. Hence TWTM is able to bridge the vocabulary gap between document content and keyphrases more precisely. Experiment results on real world dataset demonstrate that TWTM outperforms existing state-of-the-art methods under various evaluation metrics. We also demonstrate that TWTM achieves a balance between word-level projection and topic-level coverage.

Moreover, TWTM is not restricted to supervised learning. TWTM can also be adopted in unsupervised fashion. So long as we can build appropriate translation pairs to represent semantic relations between documents and keyphrases, TWTM will be able to exert its capacity. For example, for news articles, we can use news titles and contents to build translation pairs, by regarding titles as an approximate language to keyphrases; for scientific papers, we can use abstracts and contents to build translation pairs.

We design the following research plans: (1) The number of topics in TWTM requires being pre-defined by users. We plan to incorporate Bayes Nonparametric (Blei et al., 2010) for TWTM to automatically learn the number of topics. (2) The trigger probabilities in TWTM do not take rich linguistic knowledge into consideration. We plan to incorporate more complicated alignment models from SMT into our model. (3) This paper focuses on supervised learning of TWTM. We plan to investigate unsupervised learning of TWTM for documents such as news articles and scientific papers. (4) The influence of product types for keyphrase extraction has not been thoroughly investigated in this paper. We plan to study the impact of product types and explore domain adaptation (Blitzer et al., 2006) for cross-domain keyphrase extraction using TWTM.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (NSFC) under the grant No. 61170196 and 61202140. The authors would like to thank Douban Inc. for providing the anonymized data.

References

Blei, D., Griffiths, T., and Jordan, M. (2010). The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2):7.

Blei, D. and Lafferty, J. (2009). *Text mining: Classification, Clustering, and Applications*, chapter Topic models. Chapman & Hall.

Blei, D., Ng, A., and Jordan, M. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of EMNLP*, pages 120–128.

Brown, P., Pietra, V., Pietra, S., and Mercer, R. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

Buckley, C. and Voorhees, E. (2004). Retrieval evaluation with incomplete information. In *Proceedings of SIGIR*, pages 25–32.

Dempster, A., Laird, N., Rubin, D., et al. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Frank, E., Paynter, G., Witten, I., Gutwin, C., and Nevill-Manning, C. (1999). Domain-specific keyphrase extraction. In *Proceedings of IJCAI*, pages 668–673.

Garg, N. and Weber, I. (2008). Personalized, interactive tag recommendation for flickr. In *Proceedings of RecSys*, pages 67–74.

Grineva, M., Grinev, M., and Lizorkin, D. (2009). Extracting key terms from noisy and multi-theme documents. In *Proceedings of WWW*, pages 661–670.

Heinrich, G. (2005). Parameter estimation for text analysis. *Web: http://www. arbylon. net/publications/text-est*.

Iwata, T., Yamada, T., and Ueda, N. (2009). Modeling social annotation data with content relevance using a topic model. In *Proceedings of NIPS*, pages 835–843.

Krestel, R., Fankhauser, P., and Nejdl, W. (2009). Latent dirichlet allocation for tag recommendation. In *Proceedings of ACM RecSys*, pages 61–68.

Li, X., Snoek, C., and Worring, M. (2009). Learning social tag relevance by neighbor voting. *IEEE Transactions on Multimedia*, 11(7):1310–1322.

Liu, Z., Chen, X., and Sun, M. (2011a). A simple word trigger method for social tag suggestion. In *Proceedings of EMNLP*, pages 1577–1588.

Liu, Z., Chen, X., and Sun, M. (2012). Mining the interests of chinese microbloggers via keyword extraction. *Frontiers of Computer Science*, 6(1):76–87.

Liu, Z., Chen, X., Zheng, Y., and Sun, M. (2011b). Automatic keyphrase extraction by bridging vocabulary gap. In *Proceedings of CoNLL*, pages 135–144.

Liu, Z., Huang, W., Zheng, Y., and Sun, M. (2010). Automatic keyphrase extraction via topic decomposition. In *Proceedings of EMNLP*, pages 366–376.

Liu, Z., Li, P., Zheng, Y., and Sun, M. (2009). Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of EMNLP*, pages 257–266.

Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into texts. In *Proceedings of EMNLP*, pages 404–411.

Mimno, D., Wallach, H., Naradowsky, J., Smith, D., and McCallum, A. (2009). Polylingual topic models. In *Proceedings of EMNLP*, pages 880–889.

Nguyen, T. and Kan, M. (2007). Keyphrase extraction in scientific publications. In *Proceedings of the 10th International Conference on Asian Digital Libraries*, pages 317–326.

Ravi, S., Broder, A., Gabrilovich, E., Josifovski, V., Pandey, S., and Pang, B. (2010). Automatic generation of bid phrases for online advertising. In *Proceedings of WSDM*, pages 341–350.

Salton, G. and Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing and management*, 24(5):513–523.

Si, X., Liu, Z., and Sun, M. (2010). Modeling social annotations via latent reason identification. *IEEE Intelligent Systems*, 25(6):42–49.

Si, X. and Sun, M. (2009). Tag-LDA for scalable real-time tag recommendation. *Journal of Computational Information Systems*, 6(1):23–31.

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

Turney, P. (2000). Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336.

Voorhees, E. (2000). The trec-8 question answering track report. In *Proceedings of TREC*, pages 77–82.

Wan, X. and Xiao, J. (2008a). Collabrank: towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of COLING*, pages 969–976.

Wan, X. and Xiao, J. (2008b). Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of AAAI*, pages 855–860.

Witten, I., Paynter, G., Frank, E., Gutwin, C., and Nevill-Manning, C. (1999). Kea: Practical automatic keyphrase extraction. In *Proceedings of DL*, pages 254–255.