Enhancing Key Phrase Extraction by User-generated Tags

Yujing Wang
Microsoft Research
Beijing, 100080 China
yujwang@microsoft.com

Shan Jiang *
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL, 61801 USA
sjiang18@illinois.edu

Yuwei Fang *
Department of Machine Intelligence
Peking University
Beijing, 100871 China
fangyw@pku.edu.cn

Jianwen Zhang
Microsoft Research
Beijing, 100080 China
jiazhang@microsoft.com

Zheng Chen
Microsoft Research
Beijing, 100080 China
zhengc@microsoft.com

ABSTRACT

User-generated tags open up many opportunities for better organization and management of text information. In this paper, we focus on the application of key phrase extraction. We collect large amount of user-generated tags from web pages, exploit the association between words and the corresponding tags which are manually assigned to documents, and further utilize it to facilitate key phrase extraction. Benefited from the collective knowledge contributed by numerous web users, key phrase extraction can be enhanced by suggesting candidates learned from the tag cloud. We evaluate our approach on datasets of various kinds of text documents, including scientific publications, Wikipedia articles and popular web pages. Experimental results show that our method outperforms state-of-the-art unsupervised key phrase extraction algorithm, and the tag-based relevance measurement can also contribute as a feature to enhance the performance of supervised key phrase extractor.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Theory

*The work was done when the second and third author visited Microsoft Research.

WSDM 2015 Shanghai, China

Keywords

Key phrase extraction, Tagging, User-generated tags

1. INTRODUCTION

With the explosive growth of Web 2.0 platform, the huge collection of user-generated tags becomes a rich open resource which is publicly available. Users can get easy access to social tagging system and provide collaborative annotations of pictures, web pages and other kinds of web objects. For example, Figure 1 shows a web page on which provides user-generated tags associating with corresponding document. With growing popularity of the social tagging system, it offers a potentially powerful way for better organization and management of web information and the collective folk wisdom from the tag cloud opens up new opportunities for better indexing, analyzing and mining of text data.

Most of previous works considered tag suggestion and key phrase extraction as different applications and studied the methodology separately. In [15], Medelyan et al. notice that automatic tagging and key phrase extraction have natural connection with each other and investigate the performance of state-of-the-art key phrase extraction system on the tagging scenario. However, to our best knowledge, there is no previous work on exploiting the tag clouds to enhance key phrase extraction models.

In this paper, we are aiming to study how to utilize the user-generated tags to facilitate key phrase extraction. There are two major categories of key phrase extraction algorithms, namely unsupervised method and supervised approach.We argue that it is very difficult to perform key phrase extraction without supervision from human. However, state-of-the-art supervised key phrase extractors usually rely on the training samples so its effectiveness is sometimes limited to the specific language and the specific domain where the training samples are available. In contrast, user-generated tags are more general and their constant self-growth makes it potential to provide support for the construction of a universal key phrase extraction system. We can get very rich knowledge which covers multiple domains from collaboratively tagged web pages. Besides, the cost of acquisition of user-generated tags is much lower than that of special-

Figure 1: Examples of tags assigned to a news article

ly constructed training corpus. Enormous resources with user-generated tags can be obtained from the Web for free and we can easily crawl the data to build a huge tag knowledge base. Moreover, new tags or newly tagged documents are introduced everyday so tag-assistant key phrase extraction system will be adaptable for analysis of real world big data with never-ending growth. Another advantage of tag-assistant key phrase extraction lies in the pervasiveness of user-generated tags. Different from features which are only applied to specific domains (e.g., section occurrence vector for key phrase extraction in scientific literature [18]), the user-generated tags covers a wide scope of documents and tag-assistant key phrase extraction method will be applicable to various types of text data.

Specifically, we mine the correlation pattern between tags and document and further make use of it to suggest key phrase candidates. We first generate the vector presentation for both words appearing in the documents and tags which are assigned to them. The vector presentation of words are learned by the text embedding techniques on a large corpus so that it can well capture the semantics of the given words. The vector presentation of tags are obtained by aggregating the semantics from corresponding text documents. Based on the vector representation, we calculate the similarity between words and tags and construct a word-tag indexer. In this way, each word is connected to a list of tags that have the highest similarity with it, which implies its topical semantics. Another benefit of the word-tag indexer is that we can generate key phrase candidates efficiently by looking up the word-tag table. Given the word-tag indexer, the process of key phrase extraction can be divided into two steps. The first step is to get a collection of phrase candidates derived from the word-tag indexer. To cover the phrases which are not contained in the global tag corpus, we also include n-grams ($n \leqslant 3$) appearing in the original text content. Next, we need to select the most important topical phrases from the candidates. We calculate the relevance between the candidate phrases and the original document by utilizing the relevance score incorporated in the word-tag indexer. Thus, the knowledge learned by the word-tag indexer can be leveraged for improving key phrase extraction performance. Finally, the most relevant candidates can be extracted as key phrases after certain kinds of diversification.

We carry out our experiments on three document datasets covering different categories of text documents, including scientific papers, Wikipedia articles and popular web pages. The experimental results demonstrate stable improvement on all kinds of documents. Besides testing our approach in an unsupervised manner, we also evaluate supervised key phrase extractors by adding the tag-based relevance scores as additional feature, in which further improvement is observed.

The remainder of the paper is organized as follows. Related work is reviewed in Section 2. The details of the tag-assistant key phrase extraction system are introduced in Section 3. Section 4 shows the experiments. Finally we conclude our work and future work in Section 5.

2. RELATED WORK

Key phrase extraction is important to a variety of applications in natural language processing and information retrieval such as text clustering [1, 5], summarization [14, 29] and indexing [4], and has been extensively studied in recent years. There are mainly two types of methods for key phrase extraction, namely, supervised method and unsupervised method.

Supervised method cast the problem of key phrase extraction as a supervised learning task. As general supervised learning process, feature design is fundamental to supervised key phrase extraction methods. Among all the features, TFIDF is one of the most commonly used features. TFIDF is one of the most important word statistics and effectively reflects how likely a word is to occur in a random document. In some specific domains such as scientific literature, structural features are also very helpful [24]. Krulwich and Burkey use heuristics based on syntactic rules such as the use of acronyms to extract key phrases from a document [11]. In later work, syntactic patterns of key phrase candidates are broadly used in supervised methods, such as POS tag [16, 26] and suffix sequence [28, 10].

In addition to feature engineering, the usage of learning models is also significant to supervised key phrase extraction method. Early work treats the selection of key phrase as a binary classification problem and utilize different classifiers to address the problem. For instance, KEA [3, 27] employs Naïve Bayes. The problem of key phrase extraction is approached to by training a binary Naïve Bayes classifier using keyphrases and non-keyphrases in training samples as positive and negative examples respectively. Turney [23] treats documents as sets of phrases which are further classified to be positive or negative examples by Decision Tree. In [21], Sarkar et al. make comparison of performances between supervised key phrase extraction methods utilizing Naïve Bayes, Decision Trees and Multilayer Perceptron. Their evaluation results indicate that Multilayer Perceptron-based method generally works best and Naïve Bayes classifier with a suitable discretization can perform as competitive as Multilayer Perceptron. Apart from using binary classifiers to distinguish keyphrases from non-keyphrases, some previous work also models the selection of key phrase as a ranking problem. Phrases ranked higher are more likely to be good candidates and top candidates recommended by pointwise or pairwise ranking are finally selected as key phrases. Different ranking strategies can be involved in the framework. For example, graph-based ranking algorithm is used in [25] and Ranking SVM is employed in [8].

Researchers have made considerable effort to extract key phrase in an unsupervised way as well. Intuitively, the salience or importance of a candidate phrase can be reflected by its relatedness with other candidates. Deriving a graph from the documents, graph-based ranking algorithms such as PageRank [19] can be used to select important candidate phrases. In TextRank [16], which is one of the most well-recognized unsupervised methods, words are represented as vertices and co-occurrence relation controlled by the distance between them are utilized to build edges. The ranking algorithm is conducted iteratively and top ranked candidates are finally used as key phrases. Liu et al. [13] propose

a topical PageRank-based method. In their work, latent topics are first discovered by LDA and then the topical information is aggregated in topical PageRank.

Tomokiyo and Hurst propose a language model-based approach, which ranks the phrases by two features, namely phraseness and informativeness [22]. Phraseness measures to what extent a given word sequence can be considered as phrase and informativeness reflect how well a phrase captures the key idea of the document. Phraseness and informativeness are scored by pointwise KL-divergence between multiple language models, and then their scores are combined together to get the final ranking. In [6], Hasan and Ng conduct a systematic evaluation of five unsupervised key phrase extraction methods on datasets from four different domains. Their work suggests that the performance of a given method may vary on different datasets and different methods have their own strengths and shortcomings. Moreover, the way of forming phrases is of great importance to key phrase extraction and has a significant impact on the performance of the extractor.

Compared to the existing work, our method learns the association between words and tags from the Web resources and further makes use of it in both of the two steps in key phrase extraction, namely forming and ranking of the candidate phrases. However, our method does not rely on any particular training corpus and can be more adaptable to different domains, which is in line with the main advantage of general unsupervised method.

3. METHODOLOGY

Key phrase extraction typically includes two steps: (1) to generate a list of words or phrases which serves as candidate; and (2) to discriminate the appropriate key phrases from others in the candidate set [7]. Supervised method requires a collection of training data and focuses more on feature engineering to learn a classifier. The limitation of supervised approach is that the trained classifier might sometimes be dominated by the training data, which may has bias towards a certain domain. For example, if we use the structural features from a training collection of scientific literature, it is possible that the classifier may not work well on a corpus of web pages. In a sense, it is difficult to train a universal model by supervised method to recommend key phrases for all types of text data. Though the performance of unsupervised method is not always as outstanding as supervised method, unsupervised method evolves less human effort on manual annotation for training data and the extractor relies less on the specific features, which makes unsupervised method more adaptable to different domains and languages. In this paper, we aim at improving unsupervised approaches by borrowing folk wisdom from user-generated tags. Our goal is to construct a universal key phrase recommendation system which can work well on various types of text data. Moreover, in the supervised scenario, our proposed method can also be utilized as an additional feature to improve the performance of supervised key phrase extraction systems.

Our approach can be described by three majors steps. (1) Tag data collection: collecting user-generated tags from the entire web, together with the corresponding text documents; (2) Build word-tag indexer: calculate the pairwise similarity between all words and tags, and build the word-tag indexer accordingly; (3) Predict relevant key phrases given an unseen document using the word-tag indexer which we have built off-line. In the following sections, we will discuss these procedures in detail.

3.1 Tag Data Collection

Our work is inspired by the idea of recommending key phrases to documents by the assistance of user-generated tags. As a platform facilitating information sharing, collaboratively tagging system has gained its increasing popularity in recent years. There are many web sites offering author-assigned tags; or support collaboratively tagging which offers user-generated tags, e.g., news provider like NYTimes, blog web sites such as Blogger and WordPress, social bookmarking services such as Delicious and CiteULike, and forum websites such as StackOverFlow.

We can get author-assigned or user-generated tags from these web sites using general HTML patterns, which covers various domains. For some domains such as Wordpress and Delicious which are not covered by the general patterns, we collect the data by specific HTML patterns or APIs which are provided officially. Table 1 shows some HTML patterns we use for collecting <document, tag> pairs, where the "**tag**" labels with bold font stand for the position where tags can be extracted. The text content of the webpage is served as the corresponding document.

We process all popular web pages indexed by a commercial search engine, and apply the afore-mentioned HTML patterns to extract <document, tag> pairs. In this way, we collect over 10,000,000 distinct tags in the collection; and more than 300,000 tags are associated with at least 100 distinct documents. The tags we collect can function as a compressed thematic summary of the texts and each text document can be regarded as a representative for a certain tag. To ensure the quality, we finally choose those tags that have more than 100 related documents to build up our data collection. Table 2 demonstrates some examples of document-tag pairs in the collection.

Table 3, 4 and 5 demonstrate some statistics of the tag data collection. In Table 3, it shows top domains with the most document-tag pairs extracted; Table 4 lists the top tags with most relevant documents; and Table 5 lists some examples of more fine-grained tags in computer science and the number of corresponding documents we collect. We can see that there are fine-grained tags like "algorithm", "data mining" and "deep learning" which can be served as good key phrase candidates. At the same time, using the relevant documents for these tags, we can incorporate semantic representation of the tags to benefit the key phrase extraction system.

Table 3: Top domains with the most tags

domain	# of documents
pdfrepairmanual.com	8,924,787
sciencedaily.com	3,794,887
tomanual.com	3,087,557
stackoverflow.com	3,065,779
livejournal.com	1,442,189
skype.com	878,232

3.2 Construction of Word-Tag Indexer

The tags we collect can function as a compressed thematic summary of the texts and each text document can be seen as

Table 1: HTML patterns for collecting document-tag pairs

General HTML patterns for various domains	<meta *name="keywords"* content="**tag1,tag2,...**" /> **tag** **tag** <a *class="tag"* href="..." target="_blank">**tag**
Specific HTML pattern for "wordpress.com", etc.	*GA_googleAddAttr("Tag", **tag**)*

Table 2: Examples of extracted document-tag pairs

Text of Document	Relevant Tags
Raymond J. Sabata, age 74, of Kronborg, passed away Thursday, January 23, 2014, at St. Francis Medical Center. Funeral Services will be held at 10:00 A.M., Monday, January 27th at St. John Lutheran Church in Kronborg. Rev. Brian Krause will officiate. Interment will be in the St. John Cemetery. Visitation will be from 5 - 7 P.M. Sunday at Higby-McQuiston Mortuary in Aurora. Memorials may be made in care of family...	Raymond Sabata
how can we call a http servlet from mule? I have a requirement in which i need to call a servlet from mule flow.I got one answer from the below site http://mule.1045714.n5.nabble.com/Creating-query-string-for-calling-a-servlet-td2665836.html But when i copy that code in my flow it is showing some errors in flow. Is there any other method or solution to call servlet from mule. I am using mule 3.5 anypoint studio...	servlet, mule
OTTAWA – Canada Post is ending door-to-door delivery in urban areas and boosting the cost of stamps in a bid to halt a skid into financial ruin, the Crown corporation announced Wednesday. The move from door-to-door delivery to community mailboxes will be rolled out over the next five years, starting in the second half of 2014, and affect about one-third of Canadian households. The change will not affect rural households...	blame canada, canada post, mail, postal services, community mailboxes

Table 4: Top tags with most relevant documents

domain	# of documents
news	1,682,239
sports	402,015
entertainment	310,092
health	294,683
food	283,608
business	269,231
politics	247,081

Table 5: Examples of fine-grained tags

domain	# of documents
artificial intelligence	8,938
algorithm	8,043
machine learning	2,780
data mining	2,625
neural network	443
deep learning	119

a representative for a certain tag. The document-tag pairs can be served as external knowledge to provide powerful assistance to the key phrase extraction task. Our philosophy is that the association of documents and tags can be captured by the relevance between words and tags learned from the crowd; and highly associated tags can thus be recommended as key phrases. Each word can be associated with a set of tags which is most relevant to the topics it may related to. Tags that have adequate association with words in a document could be representative for the document. In other words, we utilize the association between words and

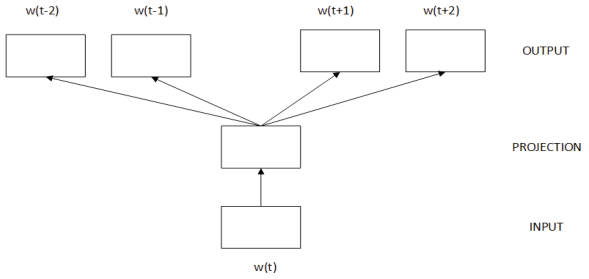


Figure 2: Framework for Learning Word Vectors

candidate tags to facilitate key phrases generation, which we named as a word-tag indexer. In the rest of this section, we will introduce the methodology to build the word-tag indexer.

3.2.1 Embedding Calculation

With the rapid development of deep learning, there have been mature techniques for training word embedding. These techniques are commonly known as neural network language models [2]. Mikolov et al. propose a fast skip-gram model *word2vec* [1] which tries to maximize accuracy of the vector operations by developing model architectures that preserve the linear regularities among words [17]. Its framework for learning the word vectors is shown in Figure 2.

In this framework, it uses each current word as an input to a log-linear classifier with continuous projection layer, and predict words within a certain range before and after the current word. In this paper, we use *word2vec* to train word

[1]http://code.google.com/p/word2vec

embedding because of its strong scalability.

We pre-train the word embedding on a large web page corpus collected from a commercial search engine. The most popular 100,000 words are used for training and the dimension of the embedded word vector is set as 300. After the training converges, words that are semantically or topically related are mapped to points that are close to each other in the vector space. For example, "Paris" and "Beijing" are close to each other, whereas "Paris" and "Food" are farther away from each other.

In the corpus of user-generated tags we collected, each tag can be assigned to multiple documents. Because the quality of user-generated tags is usually varied and low-quality tags are sometimes noisy, we only keep those with adequate recommendation to documents, which is more possible to be of high quality. In our experiment, we filter tags with less than 100 assignments and consequently there are about more than 300,000 tags left. For a given tag, we aggregate all the documents which it has been assigned to. Then the bag-of-word representation of the aggregated document is calculated and the words in the aggregated document are ranked according to their TFIDF value. Finally, top K words with the highest TFIDF value are selected as representative terms for each tag and the embedded vectors of tags are generated by aggregating the embedding vectors of each representative word. In our implementation, K is set as 50 experimentally.

More formally, let $T = \{w_i\}$ be the representative terms of a tag, and $V_i = \{v_{ij}\}$ denotes the embedding vector for word w_i. Suppose each term w_i has a weight k_i which equals to the TFIDF value from the aggregated document for a given tag, then the tag can be represented by an embedded vector $\{E_j\}$ where $E_j = \sum_i v_{ij} \times k_i$.

3.2.2 Inverted Index Construction

Given a word w and a tag t, denote their embedded vector as v_w and v_t respectively. We use the cosine similarity to measure the relevance between them:

$$Rel(w,t) = \frac{\sum_{i=1}^{n} v_{w_i} \times v_{t_i}}{\sqrt{\sum_{i=1}^{n} v_{w_i}^2} \times \sqrt{\sum_{i=1}^{n} v_{t_i}^2}}, \quad (1)$$

where n is the dimension of the embedded vector space.

As mentioned before, topically or semantically related words are mapped to vectors close to each other in the embedded vector space and the vector representation of a tag is derived from the documents it has been assigned to. As a result, high cosine similarity between a tag t and a word w usually indicates that they are strongly associated with each other from the perspective of thematic and semantic relatedness. Hence, tags with high cosine similarity to words in a document can be considered as candidate phrases for key phrase extraction. In order to make the candidate generation more efficient, we construct an inverted index which associates each word with a list of tags that are most relevant to it. Considering that tags with low similarity is not likely to be relevant to the corresponding word and may slow down the speed, we only keep top M tags in the inverted index ($M = 1000$ in our experiment).

Table 6 shows some examples from the inverted index, where words are shown in the first column and their top 6 relevant tags together with the corresponding relevance scores are shown in the second column. We can see that the relevant tags reveal the possible topics that the word might be related to. For example, "google" is associated with multiple services or products that Google provide. "Balance" can be associated with economic topics such as "trade", as well as other fields such as "PH balance" which might be relevant to chemistry or environment. In the tag vector representation of "topology", it is not only associated with explicitly related phrases such as "algebraic topology", but is also connected with topics that are latently relevant to it, for example applications such as "network design" and "distributed systems" which could be benefited from topology as a fundamental mathematical tool.

3.3 Key Phrase Extraction

3.3.1 Candidates Generation

Previous study has shown that words with high TFIDF value are sometimes quite representative for a document. If we sort all the words in a document in descending order by its TFIDF value, the top part of this ranked list can give a rough summarization of the document. In order to generate candidates for key phrase extraction, we first gather the top L words with the highest TFIDF value in a document, denoted as $W = \{w_1, w_2, ..., w_L\}$ (L is set as 30 in our experiments). Then we aggregate all the most relevant tags of each w_i in W from the inverted index we construct. The aggregated tag set T can be served as candidate set.

However, if we only use the indexed tags as the candidates, the key phrase extraction will be completely dominated by existing tags that are collected from the tag corpus we use. Moreover, the key phrase sometimes tends to be beyond the scope of the tag collection and only making use of the associated tags is not enough for candidate recommendation. Hence, we also add phrases extracted from the document to the candidate collection to address this problem. We follow the pro-processing steps as proposed in KEA algorithm [27] for candidate phrase extraction, i.e., considering all the n-grams (n $\leqslant$ 3) that do not start or end with stop word. As [9], we do not use other features such as POS tagging and shallow parsing to avoid extra slowing down of the overall performance.

3.3.2 Relevance Calculation

For each candidate phrase t in T, we calculate its relevance score with the document d as:

$$Score(d,t) = \sum_{w \in d} (Rel(w,t) + \alpha) \cdot (TF(t,d) + \beta) \quad (2)$$

where w is one of the top H ($H \leqslant 30$) key words in document d, which are selected according to TFIDF values. $Rel(w,t)$ is the relevance score of word w and phrases t, and $Rel(w,t) = 0$ if t is not in the global tag collection. α is a smoothing parameter in case that t is not in the tag collection. $TF(t,d)$ is the frequency of phrase t appearing in document d, and it will be 0 if t does not have any occurrence in the document. Parameter β controls how much the candidates which are not derived from the original document will contribute to the key phrase extraction. In the key phrase extraction scenario, we can set $\beta = 0$ if we do not want to include other tags that do not appear in the original text; while in other tag suggestion scenarios which allow topical tags without occurrence in the original document, we can set β as a non-zero value.

3.3.3 Phrase Diversification

Table 6: Example of inverted index.

Word	Relevant Tags
google	⟨google mobile, 0.3331⟩ ⟨google maps, 0.3316⟩ ⟨google earth, 0.3303⟩ ⟨google search, 0.3283⟩ ⟨google legal, 0.3185⟩ ⟨google earth news, 0.3183⟩
food	⟨canned food, 0.3909⟩ ⟨food news, 0.3892⟩ ⟨fresh food, 0.3837⟩ ⟨cheap food, 0.3722⟩ ⟨pet food, 0.3653⟩ ⟨organic food, 0.3620⟩
knowledge	⟨knowledge, 0.4067⟩ ⟨knowledge transfer, 0.3611⟩ ⟨knowledge management, 0.3603⟩ ⟨knowledge work, 0.3494⟩ ⟨background knowledge, 0.3420⟩ ⟨self knowledge, 0.3234⟩
launch	⟨quick launch, 0.3539⟩ ⟨united launch alliance, 0.3306⟩ ⟨product launch, 0.3148⟩ ⟨launching, 0.2831⟩ ⟨iphone 5 launch, 0.2686⟩ ⟨launch date, 0.2605⟩
adaptive	⟨adaptive, 0.3657⟩ ⟨adaptive leadership, 0.3460⟩ ⟨adaptive learning, 0.3415⟩ ⟨adaptive sports, 0.2799⟩ ⟨adaptive fireground management, 0.2615⟩ ⟨assistive technology, 0.2368⟩
sports	⟨youth sports, 0.3989⟩ ⟨club sports, 0.3712⟩ ⟨team sports, 0.3751⟩ ⟨women sports, 0.3554⟩ ⟨sports recruiting, 0.3515⟩ ⟨live sports, 0.3472⟩
topology	⟨topology, 0.4832⟩ ⟨general topology, 0.4177⟩ ⟨algebraic topology, 0.3734⟩ ⟨network design, 0.2661⟩ ⟨network virtualization, 0.2624⟩ ⟨distributed systems, 0.2599⟩
balance	⟨balance, 0.2923⟩ ⟨balance sheet, 0.2857⟩ ⟨trade balance, 0.2280⟩ ⟨new balance, 0.2268⟩ ⟨prepaid debit cards, 0.2121⟩ ⟨PH balance, 0.2080⟩
plant	⟨plant, 0.5521⟩ ⟨plant care, 0.5278⟩ ⟨foliage plant, 0.5016⟩ ⟨plant identification, 0.4863⟩ ⟨plant care questions, 0.4612⟩ ⟨plant propagation, 0.4541⟩
software	⟨windows software, 0.4662⟩ ⟨free software, 0.4492⟩ ⟨softwares, 0.4491⟩ ⟨internet software, 0.4389⟩ ⟨video software, 0.4385⟩ ⟨software reviews, 0.4258⟩

In the last step, tags are sorted in descending order by their relevance score, and the top N candidate phrases are recommended to users. Sometimes, the top N phrases are very close to each other, so that certain kind of diversification is needed when presenting the final results. In our method, we apply the Porter Stemmer [20] when generating the results and the duplicated phrases after stemming can thus be discarded. In real applications, we can apply more strategies to diversify the key phrase results. For example, we can remove a phrase if it is a sub-phrase of another key phrase which has better relevance score. Moreover, we can calculate the similarity of two phrases based on their corresponding embedding vectors to diversify the key phrase results. However, we do not apply these kind of diversification in our experiment because they are rather difficult to be evaluated.

4. EXPERIMENTS

We conduct systematic experiments to evaluate our proposed model on three data sets: *CiteULike-180*[2], *SOCIAL-ODP-2K9*[3] and *Wiki10+*[4]. We first evaluate our methodology as an unsupervised approach and compare to other state-of-the-art unsupervised methods. Then, we utilize our approach as an additional feature in the supervised model to examine if it can make further improvement in the supervised scenario.

4.1 Data Set

The first data set *CiteULike-180* is a collaboratively tagged data corpus extracted from the bookmarking web site CiteULike.org on June 2008 [15]. The corpus contains 180 scientific research papers, with 946 associated tags that are agreed at least by two human annotators. On average, there are about five tags per document. The agreement between annotators makes the corpus more reliable. To further avoid the potential bias caused by arbitrarily edited tags, the ground truth is constituted of documents with at least three tags annotated.

The second data set is *SOCIAL-ODP-2K9* which is collected from the Open Directory Project[31]. Open Directory Project is a multilingual open directory of World Wide Web links. This dataset is made up by 12,616 unique URLs and their corresponding text content. The user-generated tags associated with a URL is collected from the social bookmarking sites Delicious and StumbleUpon during December 2008 and January 2009. To ensure a webpage being socially popular, we limit the dataset to those pages with at least 100 users annotating them.

The third data set *Wiki10+* is created during April 2009 with data retrieved from the social bookmarking site Delicious [5] and Wikipedia [6]. This data set is made up by 20,764 unique URLs and their corresponding social tags. All of them are English Wikipedia articles with at least 10 annotations on Delicious. The tag information for each of these Wikipedia articles as well as the text content can be found in this dataset [30].

4.2 Evaluation Metrics

The evaluation is performed on the three datasets, described in Section 4.1. A predicted key phrase is considered as "correct" when it exactly matches one of the ground truth phrases after using the Porter stemmer. For example, *fuzzy cats* matches *fuzzy cat*, but it does not match either *cat* or *cats fuzzy*. We measure the performance by computing Precision (the percentage of correct key phrases out of all the key phrases extracted), Recall (the percentage of correct key phrases extracted by our algorithm out of all key phrases

[2]http://maui-indexer.googlecode.com/files/citeulike180.tar.gz
[3]http://nlp.uned.es/social-tagging/socialodp2k9/
[4]http://nlp.uned.es/social-tagging/wiki10+/
[5]https://delicious.com/
[6]http://en.wikipedia.org/wiki/

Table 7: Evaluation results on key word extraction

CiteULike-180			
	Precision	Recall	F1
TF	0.2635	0.3110	0.2853
TFIDF	0.2511	0.3022	0.2743
TextRank	0.2729	0.3298	0.2987
Tag-As	**0.2878**	**0.3319**	**0.3083**
SOCIAL-ODP-2K9			
	Precision	Recall	F1
TF	0.3378	0.1692	0.2255
TFIDF	0.3092	0.1561	0.2075
TextRank	0.3055	0.1678	0.2204
Tag-As	**0.3423**	**0.1709**	**0.2280**
Wiki10+			
	Precision	Recall	F1
TF	0.3473	0.2631	0.2994
TFIDF	0.3448	0.2579	0.2951
TextRank	0.3454	0.2607	0.2971
Tag-As	**0.3598**	**0.2692**	**0.3080**

Table 8: Evaluation results on key phrase extraction

CiteULike-180			
	Precision	Recall	F1
TF	0.2571	0.2639	0.2605
TFIDF	0.2473	0.2598	0.2534
TextRank	0.2714	0.2808	0.2760
Tag-As	**0.2824**	**0.2959**	**0.2890**
SOCIAL-ODP-2K9			
	Precision	Recall	F1
TF	0.3083	0.1682	0.2177
TFIDF	0.2845	0.1552	0.2008
TextRank	0.3108	0.1683	0.2184
Tag-As	**0.3117**	**0.1703**	**0.2203**
Wiki10+			
	Precision	Recall	F1
TF	0.3337	0.2635	0.2945
TFIDF	0.3298	0.2584	0.2900
TextRank	0.3303	0.2599	0.2909
Tag-As	**0.3428**	**0.2701**	**0.3021**

provided by the ground truth dataset) and F1-Measure:

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{3}$$

4.3 Experimental Settings

We test our algorithm on the three data sets mentioned before, namely, *CiteULike-180*, *SOCIAL-ODP-2K9* and *Wiki10+*, and perform quantitative evaluation to see how it works. In the relevance calculation function, α is set as 0.1 and β is set as 0. First, we report the results of top N key phrases when $N = 5$. Then, the performance is also evaluated with different values of N to draw a reliable conclusion.

We also compare our approach with three unsupervised baseline methods. The simplest approach is to use the term frequency (denoted as TF) to select key phrases. The top N phrases with the highest term frequency are considered as key phrases given a document. Similarly, we can also take the top N phrases with highest TFIDF value as the final key phrases recommended to users. We also employ TextRank [16], one of the best recognized unsupervised method for key phrase extraction, for a comparison.

Similar to [12], we test the key words extraction and key phrases extraction scenarios separately. The TextRank model [16] is originally designed for the extraction of single key word by representing the document as a word graph. In the multiple-words scenario, we extend the original TextRank model to a phrase graph to support key phrases extraction. In the graph, each phrase candidate serve as a vertex and the relationship is built by the distance of two phrases.

4.4 Evaluation Results

We first compare the performance of extracting key words based on different method. Note that a large part of labeled key phrases in the ground truth data set are actually composed by single words. So we first evaluate the capacity of different algorithms on extracting key words. Later we will re-conduct the experiment to extract key phrases without length limitation.

Table 7 shows the performances of the four methods on the three benchmark datasets in terms of extracting key words. Our tag-assistant method is denoted as "Tag-As" in the table, and it outperforms other methods according to all the three metrics. TextRank works better than simply using TF or TFIDF to extract key words in most cases, which is consistent with the conclusion made by previous studies [12]. It is interesting to observe that only using TF sometimes works better than using TFIDF. The reason may be that users like to use frequently appeared category terms to describe the articles, although they have low IDF values. For example, many users use the term "Wikipedia" to summarize a Wikipedia article, which indeed reflect the category from a certain perspective.

In the second experiment we employ the four different methods to extract key phrases without length limitation. The evaluation results are shown in Table 8 and the metrics used here are still precision, recall and F1-Measure. We can see that our method still works the best. The TextRank method extended for key phrase extraction is not as good as in the single-word scenario because it is not specifically designed for ranking key phrases. Besides, the performance of this scenario seems competitive to the single-word evaluation because the ground truth of all the datasets are mainly composed by single-word phrases. Therefore, to investigate how the methods perform on multiple-words phrases, we also report the evaluation results on ground truths of only multiple-words phrases. If a document does not contain ground truth phrases with multiple words, we simply skip the document in the evaluation. As shown in Table 9, our proposed method still outperforms others in the multiple-words extraction scenario.

As mentioned before, we take the top N phrases with highest ranking score (in our method $Score(d, t)$) to be the selected key phrases. To further check how different N will affect the performance, we run our algorithm with different N and show the results in Figure 3. In general, smaller N will result in higher precision but lower recall. With the growth of N, the precision declines but the recall increases. It is in line with our intuition because that the higher a phrase is ranked, the more likely that it will be a good

Table 9: Evaluation Results on multiple-words phrase extraction

CiteULike-180			
	Precision	Recall	F1
TF	0.1807	0.1426	0.1594
TFIDF	0.1687	0.1245	0.1433
TextRank	0.1687	0.1305	0.1472
Tagging	**0.2169**	**0.1787**	**0.1960**
SOCIAL-ODP-2K9			
	Precision	Recall	F1
TF	0.1372	0.1316	0.1343
TFIDF	0.1187	0.1135	0.1160
TextRank	0.1136	0.1083	0.1109
Tagging	**0.1461**	**0.1400**	**0.1430**
Wiki10+			
	Precision	Recall	F1
TF	0.4114	0.3562	0.3818
TFIDF	0.4019	0.3481	0.3730
TextRank	0.3819	0.3316	0.3549
Tagging	**0.4149**	**0.3584**	**0.3846**

candidate as a key phrase.

We also test the three baseline methods with different N. We can see from Figure 4 that our method outperforms the baseline methods and generally all the methods works best when N is set to be 5. We can find the similar trend that higher precision can be got with smaller N while higher recall can be observed when N is larger in different dataset using different methods (TF, TFIDF and TextRank) as well. Due to the space limit, we do not illustrate all the results here.

4.5 Supervised Experiments

Apart from directly using our algorithm to extract key phrase in an unsupervised way, we can also employ it as one of the features for supervised extractors to enhance the performance. Supervised key phrase extraction method generally can be divided to two steps: 1) Training: train a classifier or ranking model to score candidate key phrases from training sample set where the manually labeled golden standard are available. 2) Extraction: extract key phrases automatically from a given document by the trained extractor.

KEA is a standard key phrase extraction algorithm in state-of-the-art. It trains a Naïve Bayes classifier leveraging three features [27]:

- **TFIDF** is a measure describing the specificity of a term for this document under consideration, compared to all other documents in the corpus. Candidate phrases that have high TFIDF value are more likely to be key phrases.
- **First occurrence** is computed as the percentage of the document preceeding the first occurrence of the term in the document. Terms that tend to appear at the start or at the end of a document are more likely to be keyphrases.
- **Length** of a phrase is the number of its component words. Two-word phrases are usually preferred by human indexers.

We also take Keyphraseness as a feature, which quantifies how often a candidate phrase appears in the training corpus [15].

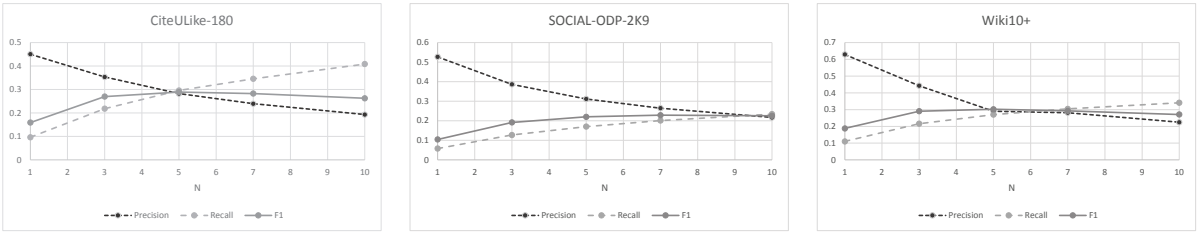
In our experiment, we use Naïve Bayes as the classifier and employ all the afore-mentioned features (i.e., TFIDF, First occurrence, Length and Keyphraseness) to train the baseline classifiers. Based on the word-tag indexer described in Section 3.2.2, we also propose a new feature denoted as *tagging* which encodes the relevance between candidate phrase and the document. More specifically, for a given document d, a candidate phrase set T is generated by the word-tag indexer. Then for each phrase t in T, we calculate its relevance score $Rel(d, t)$ with Equation 1, which is used as the tag-based feature. If t does not exists in T, the corresponding feature value can be set as 0.

We use 5-fold cross-validation to evaluate the results of key phrase extraction. The data sets are partitioned randomly into five disjoint subsets. For each time, one of the subsets is used for testing and the other four subsets are utilized as training data to train the classifier. The evaluation results are reported by taking the average of each fold. We show the results in Table 10, listing the metrics on three public datasets respectively. In the table, "TFIDF" denotes the results of only using TFIDF as feature to train the classifier; "+FirstOccurrence" makes use of both TFIDF and the first occurrence feature; "+Length" means that the length of phrase is added to the feature pool, together with TFIDF and the first occurrence feature; "+Keyphraseness" means further bringing in Keyphraseness as one of the features and "+Tagging" represents employing tag-based feature in addition to all the above features. We can see that with more features added, the performance is gradually enhanced. Keyphraseness benefits the key phrase extraction results obviously as it captures the domain knowledge for each dataset. Besides, the tag-based feature indeed helps the classifiers to extract better key phrases. As shown in Table 10, both precision and recall grow when the feature *tagging* is added, resulting in higher F1-measure. Moreover, the results of supervised classifiers outperform the corresponding unsupervised methods, which is consistent with our common sense.

5. CONCLUSIONS AND FUTURE WORK

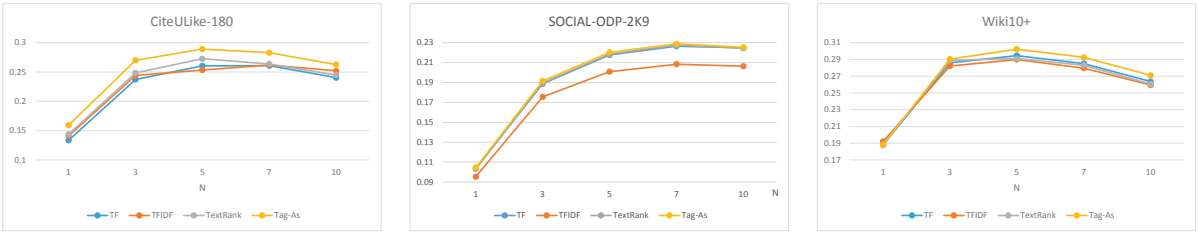
We propose a key phrase extraction method by bringing in the assistance of user-generated tags. By discovering the association between words and tags, we build an inverted indexer where each word is linked to a set of relevant tags with the corresponding relevance scores. In the candidate generation phrase, we adopt both tags from the collection as well as phrases extracted from the document itself. Then we choose candidates which are most relevant to the documents after a procedure of diversification.

Our experimental results show that the method can effectively extract key phrase from a given document. In general, candidates ranked higher are more reliable and consequently we can get higher precision with less but more refined key phrases. Besides, we can also use the tag-assistant feature to enhance the performance of supervised key phrase extraction method. Since collaboratively tagging system covers a variety of domains and languages, using the tag-assistant method can be beneficial for building a more universal key



(a) CiteULike. (b) SOCIAL-ODP-2K9. (c) Wiki10+.

Figure 3: Performance of Tag-As based on different N.



(a) CiteULike. (b) SOCIAL-ODP-2K9. (c) Wiki10+.

Figure 4: F-measure on different N of all methods.

Table 10: Results of Supervised Approach

CiteULike-180			
	Precision	Recall	F1
TFIDF	0.2865	0.2765	0.2814
+FirstOccurrence	0.2959	0.3071	0.3014
+Length	0.3068	0.3240	0.3151
+KeyPhraseness	0.3189	0.3375	0.3279
+Tagging	**0.3514**	**0.3711**	**0.3610**
SOCIAL-ODP-2K9			
	Precision	Recall	F1
TFIDF	0.2888	0.1571	0.2035
+FirstOccurrence	0.3160	0.1755	0.2257
+Length	0.3330	0.1846	0.2375
+KeyPhraseness	0.3605	0.2010	0.2581
+Tagging	**0.3761**	**0.2076**	**0.2676**
Wiki10+			
	Precision	Recall	F1
TFIDF	0.3309	0.2617	0.2922
+FirstOccurrence	0.3318	0.2660	0.2953
+Length	0.3614	0.2671	0.3071
+KeyPhraseness	0.3944	0.2562	0.3107
+Tagging	**0.4150**	**0.2717**	**0.3284**

phrase extraction system.

Although we focus on the key phrase extraction task in this paper, the tag-based feature we proposed is not restricted to it and can be utilize to facilitate various kind of applications. For example, we can leverage this feature as semantic knowledge to improve document summarization, search relevance, and Ads suggestion, which we will study in the future work.

6. REFERENCES

[1] P. G. Anick and S. Vaithyanathan. Exploiting clustering and phrases for context-based information retrieval. In *ACM SIGIR Forum*, volume 31, pages 314–323. ACM, 1997.

[2] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[3] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. 1999.

[4] C. Gutwin, G. Paynter, I. Witten, C. Nevill-Manning, and E. Frank. Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems*, 27(1):81–104, 1999.

[5] K. M. Hammouda, D. N. Matute, and M. S. Kamel. Corephrase: Keyphrase extraction for document clustering. In *Machine Learning and Data Mining in Pattern Recognition*, pages 265–274. Springer, 2005.

[6] K. S. Hasan and V. Ng. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 365–373. Association for Computational Linguistics, 2010.

[7] K. S. Hasan and V. Ng. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2014.

[8] X. Jiang, Y. Hu, and H. Li. A ranking approach to keyphrase extraction. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 756–757. ACM, 2009.

[9] S. C. Johnson and Y. Liu. Discovering contextual tags from product review using semantic relatedness. In *Proceedings of the 2013 International Conference On Engineering Design*, 2013.

[10] S. N. Kim and M.-Y. Kan. Re-examining automatic keyphrase extraction approaches in scientific articles. In *Proceedings of the workshop on multiword expressions: Identification, interpretation, disambiguation and applications*, pages 9–16. Association for Computational Linguistics, 2009.

[11] B. Krulwich and C. Burkey. Learning user information interests through extraction of semantically significant phrases. In *Proceedings of the AAAI spring symposium on machine learning in information access*, pages 100–112, 1996.

[12] S. Lahiri, S. R. Choudhury, and C. Caragea. Keyword and keyphrase extraction using centrality measures on collocation networks. *arXiv:1401.6571*, 2014.

[13] Z. Liu, W. Huang, Y. Zheng, and M. Sun. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 366–376. Association for Computational Linguistics, 2010.

[14] B. Logan and S. Chu. Music summarization using key phrases. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 2, pages II749–II752. IEEE, 2000.

[15] O. Medelyan, E. Frank, and I. H. Witten. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1318–1327. Association for Computational Linguistics, 2009.

[16] R. Mihalcea and P. Tarau. Textrank: Bringing order into texts. Association for Computational Linguistics, 2004.

[17] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Advances in Neural Information Processing Systems*, 2013.

[18] T. D. Nguyen and M.-Y. Kan. Keyphrase extraction in scientific publications. In *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers*, pages 317–326. Springer, 2007.

[19] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

[20] M. F. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137, 1980.

[21] K. Sarkar, M. Nasipuri, and S. Ghose. Machine learning based keyphrase extraction: Comparing decision trees, naïve bayes, and artificial neural networks. *JIPS*, 8(4):693–712, 2012.

[22] T. Tomokiyo and M. Hurst. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*, pages 33–40. Association for Computational Linguistics, 2003.

[23] P. Turney. Learning to extract keyphrases from text. 1999.

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

[25] X. Wan and J. Xiao. Collabrank: towards a collaborative approach to single-document keyphrase extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 969–976. Association for Computational Linguistics, 2008.

[26] X. Wan and J. Xiao. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860, 2008.

[27] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries*, pages 254–255. ACM, 1999.

[28] W.-t. Yih, J. Goodman, and V. R. Carvalho. Finding advertising keywords on web pages. In *Proceedings of the 15th international conference on World Wide Web*, pages 213–222. ACM, 2006.

[29] Y. Zhang, N. Zincir-Heywood, and E. Milios. World wide web site summarization. *Web Intelligence and Agent Systems*, 2(1):39–53, 2004.

[30] A. Zubiaga. Enhancing navigation on wikipedia with social tags. *arXiv preprint arXiv:1202.5469*, 2012.

[31] A. Zubiaga, M. Raquel, and F. Víctor. Getting the most out of social annotations for web page classification. In *Proceedings of the 9th ACM symposium on Document engineering*, pages 74–83, 2009.