Article

JIS

Journal of Information Science
2014, Vol. 40(1) 38–49

DOI: 10.1177/0165551513508877
jis.sagepub.com
SAGE

Keyword extraction for blogs based on content richness

Jinhee Park
College of Information and Communication Engineering, Sungkyunkwan University, Republic of Korea

Jaekwang Kim
College of Information and Communication Engineering, Sungkyunkwan University, Republic of Korea

Jee-Hyong Lee
College of Information and Communication Engineering, Sungkyunkwan University, Republic of Korea

Abstract

In this paper, a method is proposed to extract topic keywords of blogs, based on the richness of content. If a blog includes rich content related to a topic word, the word can be considered as a keyword of the blog. For this purpose, a new measure, richness, is proposed, which indicates how much a blog covers the trendy subtopics of a keyword. In order to obtain trendy subtopics of keywords, we use outside topical context data – the web. Since the web includes various and trendy information, we can find popular and trendy content related to a topic. For each candidate keyword, a set of web documents is retrieved by Google, and the subtopics found in the web documents are modelled by a probabilistic approach. Based on the subtopic models, the proposed method evaluates the richness of blogs for candidate keywords, in terms of how much a blog covers the trendy subtopics of keywords. If a blog includes various contents on a word, the word needs to be chosen as one of the keywords of the blog. In the experiments, the proposed method is compared with various methods, and shows better results, in terms of hit count, trendiness and consistency.

Keywords

Blogs; information retrieval; keyword extraction; LDA; subtopic model; text mining

1. Introduction

Nowadays, blogs are used as an important tool for the delivery of information and news via the Internet. Since there are usually hundreds of posts in a blog, it is important for readers to easily catch the topics of blogs. For this purpose, many approaches have been proposed, such as providing topic keywords, such as a tag cloud generated from tags in the posts or manually selected keywords by human experts, or showing categories to which those belong [1–4].

However, categories usually just give information on the domains of blogs, which is not enough to help readers to catch the content of blogs. Human experts may provide detailed and appropriate information, by selecting topic keywords from blogs that describe them well, but this needs much time and deep inspection [5]. On the other hand, a tag cloud is automatically generated based on tags assigned by authors; therefore it is widely used for blogs to provide topic keywords [6–8]. However, tags cannot always be trusted, because they are manually assigned by authors, and it is not guaranteed that every post has tags. Therefore, tags cannot be considered to reflect all of the topics in a blog [9]. Moreover, tags can be misused as a postscript of posts, or even be abused for search engine optimization, by selecting only attractive tags that are highly rated by search engines.

Corresponding author:
Jee-Hyong Lee, College of Information and Communication Engineering, Sunkyunkwan University, Seobu-ro, Jangan-gu, Suwon, Gyeonggi-do, Republic of Korea.
Email: john@skku.edu

Since topic keywords help and assist readers in understanding the overall content of posts in blogs, words that can represent the content of a blog are to be chosen to provide information about the blog. For example, let us assume that a word A has a set of subtopics $\{a_1, a_2, a_3, a_4, a_5\}$. If a blog covers content about $\{a_1, a_2, a_3, a_4\}$, it is appropriate to choose A as one of the topic keywords, because the blog covers most of the related topics about A. On the other hand, in the case of a blog including only $\{a_1\}$, A may be inappropriate as a topic keyword, because the blog does not cover subtopics of A well. When we choose topic keywords, we need to consider the coverage of keywords. The coverage is how well a keyword covers the content of a blog. However, the coverage is not a sufficient condition for topic keywords. Another is the trendiness. If readers look for blogs related to a topic keyword A, they may expect content that is not only strongly related to A, but that also fits with the current public interest in A. Readers will be more satisfied with such blogs.

There are many approaches for the extraction of topic keywords from documents, such as word-graph-based, link-based and probability-based approaches [1, 10–19]. Word-graph-based approaches, such as *TextRank* [12] and *HITS* [13], mainly consider the frequencies of words in documents, without considering the coverage or trendiness. These methods choose a word as one of the topic keywords if the word frequently appears together with important words. These methods are appropriate for choosing topic words in single documents.

There are also approaches based on the link structure between blogs [14–16]. Since those approaches consider blogs together with other neighbouring blogs, the subtopic coverage and trendiness can be implicitly reflected into chosen topic keywords. However, blogs have a tendency to be tightly connected to blogs with similar contents, so the topics may be too narrow or specific. Additionally, there are no common link architectures between blogs to be easily crawled or identified. This can be one problem in generalizing these methods.

Other research selected topics based on probabilistic analysis [17–19]. Latent Dirichlet allocation (LDA) is widely applied, because it is appropriate for topical analysis. These approaches are based on their own probabilistic models, extended from the original LDA model. However, these models have a higher complexity than the original LDA model. Furthermore, these approaches are dependent on specific information, such as citations or comments, which is difficult to generalize for every blog.

In this paper, a method is proposed to extract topic keywords of blogs, considering the coverage of trendy subtopics of keywords, by a probabilistic analysis based on LDA. In order to obtain trendy subtopics of keywords, we use outside topical context data, the web. If we collect the documents on a keyword, t, from the web, the documents may include various and trendy subtopics of t. From the web documents on t, we build a probabilistic subtopic model of t. Since subtopic models are built from the web, the models are called the web context model of t. Based on web context models, a new measure is proposed: namely Richness, which indicates how much a blog reflects the web context, by measuring the coverage of the blog on the trendy subtopics that can be found in the current web. The proposed method selects keywords based on the richness measure, so that words that represent the trendy and diverse contents in a blog are chosen.

Most existing approaches choose a word if it covers the content of documents. However, we choose a word if the document covers most aspects of the word. This ensures that the meaning of keywords overlaps well with the content of documents.

The rest of this paper is organized as follows. Section 2 reviews the related work in the topic keyword selection. Section 3 describes the proposed topic keyword selection approach. Section 4 presents the experimental results. The paper is summarized and concluded in Section 5.

2. Related work

For the keyword extraction from a set of documents, several approaches have been proposed, such as word-graph-based, link-based and probability-based approaches. Word-graph-based methods approach the problem by building graphs of words. The connection weights between them can be determined by the distance between the documents, or the count of co-occurrences. In order to determine important nodes in a graph, graph analysis methods are applied to word-graphs, such as *PageRank* and *HITS*.

Mihalcea and Tarau proposed *TextRank* [12], which is a representative keyword extraction approach based on *PageRank*. In *TextRank*, nodes are terms in a document, and links between nodes are bi-directional, because there are no directions between co-occurring terms, while the original *PageRank* algorithm uses directed hyperlinks between web documents. Hyperlink-Induced Topic Search [13] (*HITS*) is a graph analysis method, which was originally proposed for web document analysis. It divided documents on the web into hubs and authorities. An authority was semantically defined as a document that provided useful information for readers, and a hub was defined as a document that provided links to authorities. It calculated hub scores and authority scores for each document, by an iterative process. If *HITS* is applied to a word-graph, keywords can be chosen, based on either hub scores or authority scores.

The word-graph-based methods are usually used for a single document, not for a set of documents. The method analyses the structure of a word-graph of a document, rather than considering the topical context in a set of documents. For this reason, keywords may be chosen from a local viewpoint.

Chen et al. suggested an analysis method for topic trends from the network of several blogs connected with common interests [14]. They also predicted the topics to be discussed in the future in a blog or a community, using their blogging-behaviour model in a supervised manner. The model is based on the graph representation of bloggers with temporal information. Qamra et al. also applied a community-based approach with temporal clustering, to find shared interests to identify topics and keywords [15]. Sekiguchi also used a similar approach. They extracted shared interests from communities of bloggers, and identified the topics from each blog, based on the common interests. They calculated a similarity distribution on words to evaluate topic scores [16].

Since those approaches analysed blogs together with other neighbouring blogs, trendy subtopics in a set of blogs could be implicitly considered for selecting topic keywords. However, blogs are usually tightly connected only to blogs with similar topics, and rarely have connections to blogs with different topics. The group of connected blogs based on the link structure usually contains only quite similar blogs, so their shared interests may be biased or specific. In other words, some words that are unimportant but frequently used by bloggers may be overestimated. Additionally, these approaches need additional information, such as the temporal metadata, and the link structure between blogs, which are not always available. Blogs usually have few links to other blogs, and may not provide machine-readable temporal metadata. It is not easy to identify the link structures between blogs, because there are no common or standard link architectures.

Some other researches approach the problem with probabilistic models. Nallapati and Cohen suggested a method to find the topic-specific influences of blog posts, by analysing citations between blog posts using machine learning techniques [17]. They suggested a model named Link-PLSA-LDA. They grouped blog posts into two groups, 'cited' and 'citing', and built a bipartite graph by citations, because citations were a good indicator of influences. Their model also considered the content of blog posts. Ahmed and Xing analysed blog posts from a perspective of ideology, using topical analysis by multi-view LDA [18]. They assumed that the contents of blog posts were affected by the writers' ideological beliefs and the background topics of each ideology, so they added some more steps of the generation of each word in a document to the original LDA generative model. The study of Yano et al. introduced a comment prediction method from political blog posts, by applying LDA on blog posts [19].

In these studies, the LDA model was extended to their own generative models of blogs, which had a higher level of complexity than the original LDA model. They were also dependent on specific information, such as citations or comments, which not all blogs provide. The subtopic coverage could be reflected, because the LDA decomposes blogs into several subtopics. However, they analysed only the given blog, without considering outside topical context. The subtopics may be found in a local viewpoint, and thus the subtopic coverage may not be complete.

In addition to these approaches, there are some studies based on word frequencies. They usually measure the importance of each word using co-occurrences or statistical models [1, 10, 11]. However, these methods do not consider various aspects of words, so the chosen keywords may cover the content of documents in limited aspects. There are also some other types of research that use external data such as a thesaurus or ontology [20–22]. A thesaurus and ontology can be a good source as a secondary knowledge for extracting keywords, but they may carry a high cost of construction.

3. Proposed method

3.1. Overview

In this paper, a new measure, *Richness*, is proposed. The richness of a blog for a given keyword is a score that indicates how much trendy content the blog includes, from the viewpoint of the given keyword. In order to obtain the trendy subtopics of a keyword, the web context related to the keyword is extracted. The content of blogs is evaluated, based on the web context.

The proposed method is based on an assumption that a topic includes several subtopics. For example, the subtopics of 'Smart Phones' can be 'iOS', 'Android', 'Apps', etc. If a topic keyword is given, its subtopics are identified. Next, how much the content of a blog is related to each subtopic is evaluated. If a blog poorly covers the subtopics of a keyword, then it is regarded that the blog has poor content on the keyword. On the other hand, if a blog covers most of the subtopics well, it is regarded as rich. For example, let us assume that a topic keyword A has a set of subtopics $\{a_1, a_2, a_3, a_4\}$, and another keyword B has a set of subtopics $\{b_1, b_2, b_3, b_4\}$. If a blog covers $\{a_1, a_3, a_4, b_2\}$, then keyword A may be preferred over B. The blog covers more subtopics of A than B.

When evaluating the subtopic coverage, we extract the current trendy subtopics of a given keyword. For this, the web documents related to a keyword are collected, using web search engines such as Google, and a method that can discover

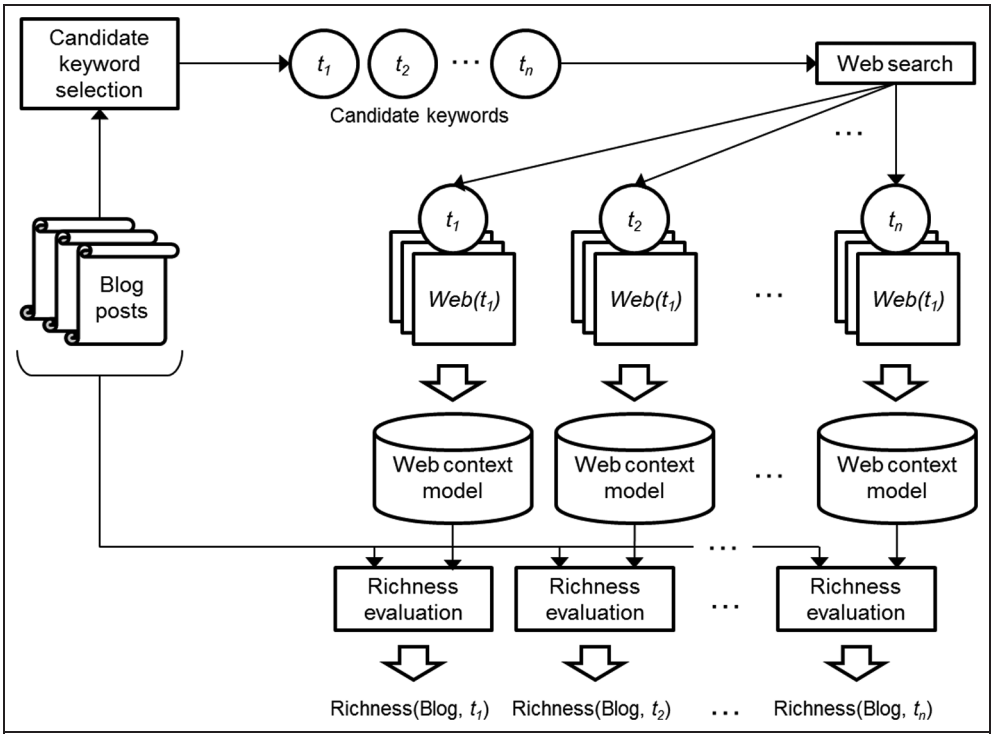

Figure 1. Overall processes of the proposed method.

hidden topics in documents is applied, such as LDA. The web contains various documents that reflect the current interests of users. The subtopics found in documents collected from the web can be considered as trendy ones. In order to model trendy subtopics, we build a topical model, by applying a probabilistic method to the documents from the web. Since the model is built from the web document, it is called a web context model.

In order to evaluate the subtopic coverage of a blog for a keyword based on the web context, the probabilities that each post of the blog is generated from each subtopic of the web context are calculated. These probabilities are considered the similarity between subtopics and blog posts, because a high probability means that the content of a post and the context of a subtopic are highly related. The higher the probability is, the more similar the content of the posts is to the web context. The richness of a blog on a keyword is calculated, based on the similarities of blog posts, and the words on which a blog has a high richness are selected as keywords. In summary, if the content of a blog is highly related to the web documents on a topic word, we regard the blog as having rich content on the topic, and the word is chosen as a keyword.

Figure 1 depicts the three steps to extract the topic keywords of a blog based on the content richness. First, the candidate keywords are chosen in all the words appearing in a blog. The candidate keywords are the words that contribute much to the content of a blog, and are thus important to the blog. Then, the web documents for each candidate keyword are collected. Using the collected web document, the web context models of each candidate keyword are created. The richness of a blog for a candidate keyword is calculated based on the web context models, and the candidate words are ordered, according to the richness. The top-ranked words are chosen as the keywords of the blog.

3.2. Latent Dirichlet Allocation

In this subsection, we briefly describe LDA, which the proposed method adopts to extract subtopics from the web context. LDA is a probabilistic generative model for a set of documents based on the bag-of-words assumption [23]. It is a statistical method to identify latent topics from observable data in documents. It is often used in the information retrieval area, for topical analysis of a document collection.

Figure 2 depicts a generative model of documents in LDA. In the figure, w indicates a word; z is the topic of word w in document d; θ follows the Dirichlet distribution with parameter α, and determines the topic distribution for document d; β is the parameter of the Dirichlet prior, which determines the probabilities that those words belong to topics. For the generation of document d based on the model, a topic distribution, θ, is determined by the Dirichlet distribution with

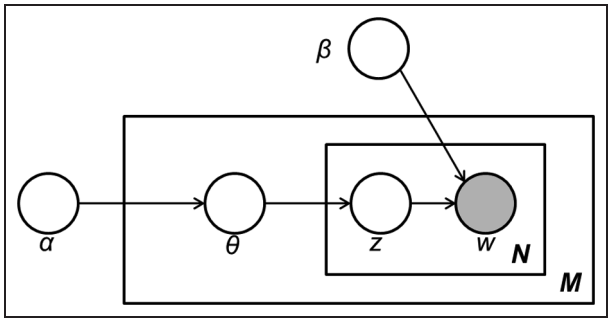

Figure 2. Generative model of LDA.

parameter α, and the number of words, N, is determined by the Poisson distribution. For each of N words, the topic z is selected by a multinomial distribution with θ, and w is selected by a multinomial distribution conditioned by z and β.

If documents are generated by such a model, and words in documents are observable, we can infer the hidden parameters θ and z from the given parameters α and β, using Bayesian inference, as follows:

$$p(\theta, z|w, \alpha, \beta) = \frac{p(\theta, z, w|\alpha, \beta)}{p(w|\alpha, \beta)} \quad (1)$$

For the inference, various approximation methods, such as Laplace approximation, Markov chain Monte Carlo or collapsed Gibbs sampling, are used, because it is generally intractable.

3.3. Candidate keyword selection

The candidate keywords of a blog are chosen from the words appearing in the blog. First, the candidate scores of all the words in a blog are evaluated, based on how much each word contributes to the content of the blog. Since keywords of a blog are strongly related to the important topics of a blog, we apply a topical analysis method, LDA, to the posts in a blog. LDA will topically cluster the posts in a blog, and gives the probabilities that each word will appear in each cluster. Since a cluster represents a topic of a blog, if a word has a high probability to a cluster, it means that the word contributes much to the content of the blog. Therefore, we can regard words with a higher probability as being more important to a blog. The candidate score of each word is evaluated, based on the probabilities.

In order to apply LDA, the number of clusters needs to be given. Usually, it is determined by experts. In the proposed method, it is chosen in an automatic way. We assume that the number of topics in a blog is proportional to the number of unique words in the blog. The number of unique words in a set of documents on a certain topic would not increase just because the number of documents increases. There would be important words for each topic, and those words would be repeated in the documents on the same topic. Therefore, it is reasonable that the number of the unique words in a document set would be more proportional to the number of topics in the documents than to the number of documents. Based on statistical analysis, we assume that a topic is composed of at least 1000 unique words. Therefore, at the candidate word selection step, we set the number of clusters by dividing the number of unique words in a blog by 1000.

Figure 3 illustrates an example of topic clusters of a blog by LDA. The posts in a blog are grouped into n topical clusters. The probability that w_i belongs to z_k, $p(w_i|z_k)$ can be interpreted as the contribution of w_i to the topic represented by z_k. Based on these probabilities, we evaluate the candidate score for w_i. We try two different approaches for the candidate score of w_i: $CS_s(w_i)$ and $CS_m(w_i)$.

Let us assume that there are m blog posts $\{b_1, b_2, \ldots, b_m\}$, n clusters $\{z_1, z_2, \ldots, z_n\}$ and v unique words $\{w_1, w_2, \ldots, w_v\}$. If LDA is applied to posts, we can obtain $p(b_i|z_k)$ and $p(w_j|z_k)$. In CS_s, the candidate score of a word is calculated by accumulating the probabilities that the word belongs to all the clusters found by LDA, as shown in equation (2).

$$CS_s(w_i) = \sum_{k=1}^{n} p(w_i|z_k) \quad (2)$$

Since $p(w_i|z_k)$ can be considered as the contribution of w_i to z_k, the accumulated value over all the clusters can be considered to be the importance of w_i in a blog.

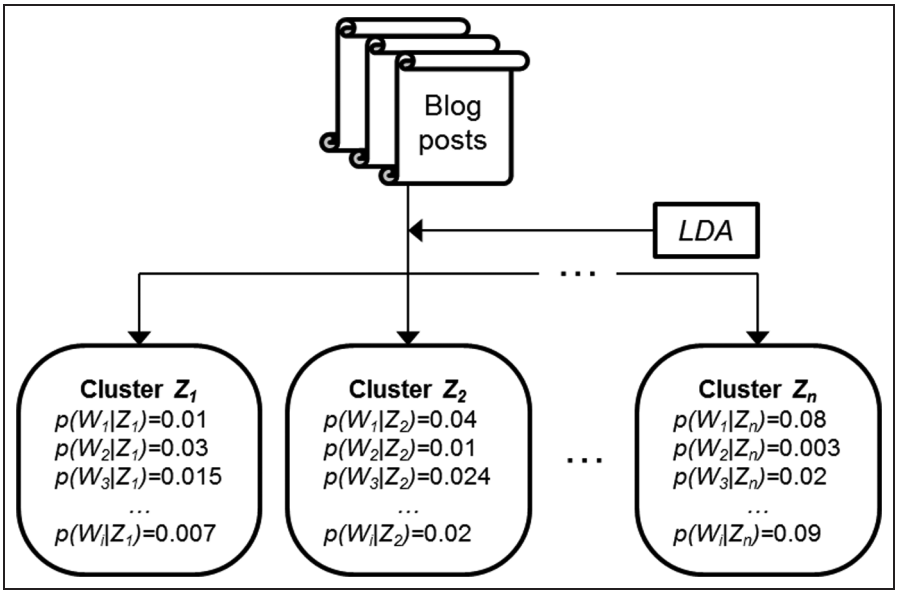

Figure 3. Topical analysis of a blog for candidate keyword selection.

However, this approach has a potential problem. Some unnecessary keywords may have high probabilities in some clusters. For example, in a tech blog, the word 'installation' may appear in most of the clusters. If its probabilities are summed up over every cluster, its candidate score can be higher than that of 'App', which can be more important than that of 'installation'. Therefore, unnecessary keywords can be selected as candidates. To address this problem of CSs, the score is calculated with a different approach, as shown in equation (3):

$$CS_m(w_i) = \max_{k=1}^{n} p(w_i|z_k) \tag{3}$$

In CS_m, the maximum is selected among the probabilities that a word belongs to each cluster, and it is regarded as the candidate score. If a word contributes strongly to at least one cluster, it can be chosen as a candidate keyword. Since the maximum value is chosen, the number of clusters in which a word appears does not affect the candidate score; 'App' may have a higher score than 'installation'.

3.4. Web context model and richness

After candidate keywords are chosen, we identify the trendy subtopics of each candidate keyword, using the outside topical context. One of the best outside topical contexts is the web, because it includes various documents that reflect the current interests of users. In order to obtain the subtopics of a candidate keyword, we collect web documents related to the keyword, using web search engines, analyse the content of the collected documents and extract the subtopics for the keyword. Since documents collected from the web may include various content that currently attracts public interest, the subtopics found in the documents are important and trendy ones of the given candidate keyword. For convenience, the set of web documents that are collected from the web related to keyword t is denoted as $Web(t)$. In order to model the subtopics appearing in $Web(t)$, we also apply LDA to $Web(t)$. The probabilistic model of subtopics for t is also called the web context model of t.

As shown in Figure 4, the web context model of t also consists of topical clusters. The number of topical clusters is also determined by the number of unique words in $Web(t)$, as for the candidate keyword selection. In the figure, s_i denotes the ith subtopic found in $Web(t)$. If we apply LDA, we can obtain the probability that a word w is included in one subtopic of t, $p(w|s_i)$, and the probability that a subtopic of t is included in $Web(t)$, $p(s_i|Web(t))$.

For a blog B and a keyword t, the probability that a post, b, in B is created under s_i, a subtopic of $Web(t)$, can be evaluated as shown in equation (4):

$$p(b|s_i) = \prod_{w \in b} (p(w|s_i))^{TF(w,\,b)} \tag{4}$$

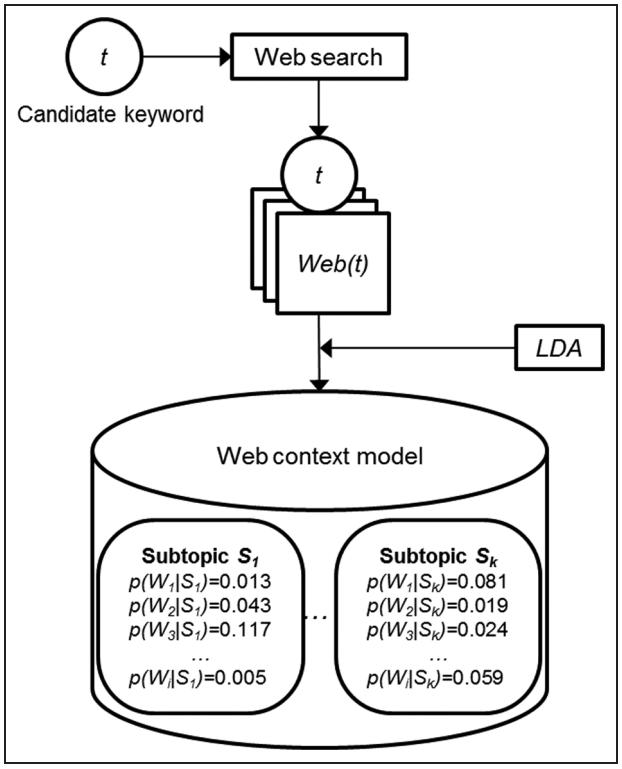

Figure 4. Building the web context model for a candidate keyword.

In the equation, $TF(w,b)$ is the frequency of w in b. The probability, $p(b|s_i)$, can be considered as the coverage of a blog post, b, to a subtopic, s_i. As b is composed of more important words in s_i, the probability will be higher, and it is more probable that b covers the contents of s_i.

When calculating $p(b|s_i)$, there is a problem that not all the words in b may appear in $Web(t)$, because $Web(t)$ and b have different vocabulary sets. If a word v in b does not appear in $Web(t)$, $p(v|s_i) = 0$. This may result in $p(b|s_i) = 0$. In order to avoid this case, the minimum value of $p(w|s_i)$ for w in $Web(t)$ is used as $p(v|s_i)$, if v does not appear in $Web(t)$.

Based on values of $p(b|s_i)$, we can calculate the probability that a blog B is generated from s_i, $p(B|s_i)$. It can be calculated based on the probabilities that each post in B is generated from s_i, as shown in equation (5):

$$p(B|s_i) = \prod_{b \in B} p(b|s_i) \tag{5}$$

This probability represents a type of correlation between B and s_i. However, we cannot use this as a coverage measure, because $p(B|s_i)$ decreases, as the number of blog posts increases. As the number of blog posts highly related to s_i increases, the coverage of B on s_i should increase. Therefore, the probabilities of blog posts are summed up, to make the coverage score increase, as the number of blog posts increases, as shown in equation (6):

$$\text{Coverage}(B, s_i) = \sum_{b \in B} p(b|s_i) \tag{6}$$

Based on this subtopic coverage of B, we define the richness of B on keyword t, as follows:

$$\text{Richness}(B, t) = \sum_{s \in S} \text{Coverage}(B|s) p(s|Web(t)) \tag{7}$$

In the equation, S is the set of subtopics of t found in $Web(t)$. The richness of B on t is the weighted summation of the coverage of B on all subtopics of t. In the equation, $p(s|Web(t))$ is the weight, which is the probability that a subtopic of t is included in $Web(t)$. Therefore, the richness of B on t will increase, as B covers many important subtopics of t. If the richness of B on t is high, that is, if B has many posts on important subtopics of t, we can say that t is one of the

Table 1. Examples of true keywords.

Blog	Keywords
Engadget	Blackberry, Bluetooth, call, case, Dell, Engadget, Ericsson, EVO, fanboy, Galaxy, graphic, hardware, HD, HTC, laptop, Motorola, nVidia, Optimus, perform, power, program, PSP, RAZR, review, Sony, Tegra, Thinkpad, touch, TV, Ubuntu, Vaio, version, video, WiFi, WinMo, XNA, Zune, ZuneHD
MacRumors	air, amazon, application, Arrandale, carrier, CDMA, chipset, eBook, feature, geForce, GSM, hackintoshi, IGZO, iMac, iTunes, Jobs, LCD, leopard, lion, LTE, MacWorld, Merom, Nehalem, new, OS, Penryn, PowerPC, Radeon, release, retina, Safari, server, SIM, snow, Steve, store, tablet, upgrade, Verizon, WWDC
The Next Web	Baidu, brand, cloud, content, Digg, Dropbox, eBay, event, Facebook, Foursquare, Geeknet, Groupon, handset, Instagram, internet, Lumia, media, microblog, Netflix, news, online, patent, Paypal, service, smartphone, social, Spotify, startup, storage, tech, think, tweet, Twitter, user, Viber, Web, Weibo, Yahoo

important topics of B. For keyword selection, we evaluate the richness of B on each candidate keyword, and arrange the candidate keywords, according to richness scores. The top N words will be recommended as the keywords of B.

4. Experiments

4.1. Experiment setup

For the experiment, Technorati,[1] which is a popular meta-blog site, was used. We chose 14 blogs listed in Technorati, and collected 1000 posts from each blog. The result was compared with a *TF-IDF*-based method, a pure LDA-based method, and three word-graph-based methods: *TextRank,* $HITS_{hub}$ and $HITS_{auth}$. The *TF-IDF*-based method simply chooses keywords from a blog based on the *TF-IDF* measure. In the pure LDA approaches, LDA was applied to the posts of a blog, without considering the web context information. It was equivalent to our candidate selection approaches: CS_s and CS_m. LDA_s is a pure LDA, which simply accepts all the candidate keywords chosen by CS_s; and LDA_m is a pure LDA, based on CS_m. We compared these, to verify the effect of the outside topical context. The proposed richness-based keyword selection approaches were $Rich_s$ and $Rich_m$. $Rich_s$ selects keywords among the candidate words chosen by CS_s, and $Rich_m$ selects among those by CS_m. *TextRank* is the keyword selection by *TextRank*. HIT_{hub} and $HITS_{auth}$ are based on the hub scores and the authority scores of *HITS*, respectively. We hardly compared our method with link-based and other probabilistic approaches [14–19], because they are based on special features, such as links between blogs, or citations between posts, which are not included in our dataset.

For the *TF-IDF*-based method, we merged all the posts in a blog into one large document, because *TF-IDF* is usually applied to a set of single documents. We built up 14 documents from the 14 blogs. Then, the traditional *TF-IDF* method was applied to the documents, and the keywords of each document were extracted. These were regarded as the keywords of the corresponding blogs. For the word-graph-based approaches, we first evaluated the score of a keyword for each post, by applying a word-graph-based method to each post, and obtained the score of a keyword for a blog, by summation of those scores. Keywords were chosen that had high scores for a blog. For the proposed approaches, we needed web documents to evaluate the richness of candidate keywords. In the experiments, 10 web pages were downloaded from Google for each candidate keyword.

The results were compared with the keywords manually selected by human experts. Two graduate students selected candidate keywords from each blog and the intersection of both candidate sets was used as the true keywords. For example, the true keywords for three blogs, Engadget, MacRumors and The Next Web, are presented in Table 1.

For the comparison, hit counts, NDCGs, trendiness and consistency were evaluated. The hit count is the number of true keywords in the recommended keywords. When evaluating hit counts, it was not considered how strongly the true keywords were recommended, that is, their ranks in the recommended keyword list. If a true keyword was in the recommended list, the hit count increased by one. In contrast, when evaluating NDCGs, the ranks of the true keywords in the recommended list were considered. NDCGs are the Normalized Discounted Cumulative Gains, which show how many true keywords appear in high positions of the recommended list. The trendiness of a keyword was evaluated based on Google Trends.[2] If a keyword is related to topics that currently attract public interest, the query volume of the keyword will increase. So, if a keyword has a high query volume, it can be regarded as being related to trendy topics. We evaluated the trendiness of keywords based on the relative query volumes obtained from Google Trends. Finally, the standard deviation of hit counts over the 14 blogs was calculated, to evaluate the consistency of methods. If the standard

Table 2. Hit count comparison.

N	$Rich_m$	$Rich_s$	LDA_m	LDA_s	*TextRank*	$HITS_{hub}$	$HITS_{auth}$	*TFIDF*
10	5.5	5.3	5.4	3.3	4.1	4.6	4.3	2.5
20	10.1	10.1	9.4	6.6	7.4	8.4	7.9	5.0
30	15.0	13.4	14.1	9.7	10.9	12.9	11.5	6.8
40	18.9	16.2	18.7	13.7	14.8	16.1	14.3	9.1

deviation of a method is lower, it means that the method shows a consistent performance for all the blogs. A high deviation is not desirable for a method, even though its average performance is high.

4.2. Results

The hit count is the number of true keywords in the recommended keywords by each method. Table 2 shows the results. The values are the average hit counts over 14 blogs. In the table, N is the number of recommended keywords by each method: N = 10, 20, 30, and 40. For example, in the case of N = 10, where each approach recommends 10 keywords, there are 5.5 true keywords by $Rich_m$, 5.3 true keywords by $Rich_s$, 5.4 true keywords by LDA_m, etc. $Rich_m$ shows the best performance for all the cases. The richness-based methods, $Rich_m$ and $Rich_s$, outperform *TextRank, HITS* and *TFIDF*. Also, $Rich_m$ and $Rich_s$ are superior to their corresponding pure LDA-based methods; $Rich_m$ is superior to LDA_m, and $Rich_s$ is superior to LDA_s. This shows that the richness scores work effectively in selecting keywords from candidate sets.

The hit counts are for comparison, in terms of the number of true keywords. For comparison, in terms of the recommendation quality of each true keyword in the recommendation list, the results were compared by NDCGs. Let us consider two cases where a true keyword appears at the top of the recommended list, and where the same keyword appears at the bottom of the list. If we consider the hit count, the hit count will increase by one, in both cases. However, if we consider the recommendation quality, the former case is better, because the true keyword is more strongly recommended than the latter case.

NDCGs are a measure indicating the appropriateness of a sequence of elements, when the true sequence is given:

$$\mathrm{NDCG} = \frac{\mathrm{DCG}}{\mathrm{IDCG}} \tag{8}$$

$$\mathrm{DCG} = rel_1 + \sum_{i=2} \frac{rel_i}{\log_2 i} \tag{9}$$

Equation (8) shows the computation of NDCG, which is a normalized DCG. DCGs are defined as shown in equation (9). In the equation, rel_i is the usefulness of the ith item in the list, and it is discounted by $\log_2 i$. The usefulness of an item is a score relative to the position in the true list. The top-ranked items in the true list have a high usefulness. If items with a high usefulness are located at the top of a recommended list, the DCG becomes higher. In order to remove the effect of the number of true keywords in recommended lists, we normalize DCG by IDCG (the ideal DCG). The ideal DCG is the DCG of a list, where every true keyword is sorted by its usefulness score. By dividing DCG by IDCG, we can remove the effect of the number of true keywords [24].

For evaluation of NDCGs, the usefulness of the ith item should be defined. In this paper, the usefulness is defined as shown in equation (10). If an item is in the true keyword list, then the usefulness is 1, otherwise it is 0. The true keyword lists given by experts are sets of keywords, not a sequence of keywords, so we simply define the usefulness, depending on whether a keyword is in the true keyword list, or not.

$$rel_i = \begin{cases} 1 & \text{if } t_i \in \text{true keyword list} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

Table 3 describes the results of NDCG comparisons. The scores range from 0 to 1, because they are normalized. According to the table, $Rich_s$ has the best performance, and LDA_m is next. Except for the case of N = 10, $Rich_m$ shows a good performance. This indicates that the proposed method successfully recommends true keywords into higher positions; the proposed methods provide better recommendation qualities. Comparing $Rich_m$ and $Rich_s$, $Rich_m$ recommended more true keywords than $Rich_s$, as shown in Table 2, but the NDCG values of $Rich_m$ are a little lower than those of

Table 3. NDCG comparison.

N	$Rich_m$	$Rich_s$	LDA_m	LDA_s	TextRank	$HITS_{hub}$	$HITS_{auth}$	TFIDF
10	0.78	0.84	0.88	0.87	0.78	0.88	0.74	0.69
20	0.76	0.80	0.79	0.69	0.69	0.73	0.71	0.75
30	0.79	0.80	0.79	0.68	0.70	0.73	0.72	0.80
40	0.80	0.80	0.79	0.69	0.70	0.73	0.73	0.64

Table 4. Trendiness comparison: DCGs based on Google Trends.

N	$Rich_m$	$Rich_s$	LDA_m	LDA_s	TextRank	$HITS_{hub}$	$HITS_{auth}$	TFIDF
10	13.73	14.29	12.88	6.48	9.18	9.63	8.58	6.03
20	24.89	25.30	22.14	14.69	16.59	19.08	17.27	16.56
30	37.64	33.57	34.45	22.85	23.82	30.84	25.97	18.71
40	46.74	40.90	45.48	32.13	33.20	38.21	33.99	28.28

Table 5. Consistency comparison: standard deviations of hit counts.

N	$Rich_m$	$Rich_s$	LDA_m	LDA_s	TextRank	$HITS_{hub}$	$HITS_{auth}$	TFIDF
10	1.24	1.87	2.50	1.91	1.98	2.31	1.91	1.73
20	2.23	3.60	3.58	2.75	3.00	3.63	3.29	3.49
30	2.34	4.29	4.66	3.17	3.78	3.94	3.92	4.57
40	3.29	5.41	5.45	3.87	4.86	4.96	4.83	5.79

$Rich_s$, which means that the true keywords are a little lower ranked in the recommended list by $Rich_m$ than in the list by $Rich_s$. It is interesting that the NDCG values of LDA_s are very low, but those of $Rich_s$ are the highest.

The trendiness is measured based on the relative query volumes from Google Trends. If a keyword is in the current public interest, it will be queried. Therefore, if a keyword is often queried, it can be regarded as popular and trendy. We evaluate how many trendy results a method produces, by evaluating how often the keywords recommended by the method are queried. The relative query volumes are collected for all of the keywords recommended by all of the methods. In order to evaluate the trendiness, we use DCGs rather than NDCGs, because the trendiness should reflect both the number of true keywords and their positions in recommended lists. We define the usefulness score of t as shown in equation (11):

$$rel_t = \log GTrend(t) \tag{11}$$

$GTrend(t)$ is the relative query volume of t by Google Trends. Since it is known that the query amount of words follows Zipf's law, we apply the log function to $GTrend(t)$ [25]. The result is shown in Table 4. In the table, $Rich_s$ shows the best result when $N = 10$ and 20, and $Rich_m$ is the best when $N = 30$ and 40, which shows that our richness scores are very effective in extracting trendy keywords from blogs. Interestingly, LDA_s shows very bad results, but $Rich_s$ shows very good results. The richness helps $Rich_s$ to recover the bad performance of LDA_s.

Finally, the consistency of the proposed method is compared with other methods. The consistency can be measured by the standard deviation of the hit counts of the 14 blogs in the experiments. Table 5 shows the consistency of each method.

A lower standard deviation indicates that a method consistently shows good results over all the blogs. On the other hand, if the deviation is high, this means that a method shows good results on some blogs, but bad results on some other blogs. From the viewpoint of consistency, $Rich_m$ also shows the best result. The performance variation over blogs is the lowest, which means that $Rich_m$ stably shows a good result for all of the blogs. LDA_m, which showed a similar performance to $Rich_m$ in the average hit count, has the worst consistency scores. This is also evidence that the richness score is effective for choosing keywords.

Table 6. *RankScore* comparison.

N	$Rich_m$	$Rich_s$	LDA_m	LDA_s	*TextRank*	$HITS_{hub}$	$HITS_{auth}$	*TFIDF*
10	2.25	2.75	3.50	5.50	5.50	4.00	5.75	6.50
20	1.75	2.50	3.50	5.75	5.50	5.25	5.00	6.25
30	1.50	3.25	3.75	6.00	5.50	4.50	5.00	6.00
40	1.00	3.25	3.50	5.75	5.25	4.25	4.50	8.00

In order to compare the overall performance, we define *RankScore* as follows:

$$RankScore(m) = (RH(m) + RN(m) + RT(m) + RC(m))/4 \quad (12)$$

In the equation, m is a keyword selection method, and $RH(m)$, $RN(m)$, $RT(m)$ and $RC(m)$ are the performance rank of m in the hit count, NDCG, trendiness and consistency, respectively. For example, if $N = 10$, $RH(Rich_m) = 1$, $RN(Rich_m) = 5$, $RT(Rich_m) = 2$ and $RC(Rich_m) = 1$, because $Rich_m$ is the best in hit counts, the fifth in NDCG, the second in trendiness, and the best in consistency. Table 6 shows the *RankScore* of each method.

In the comparison of *RankScore, Rich*$_m$ is the best, $Rich_s$ is next and LDA_m is third. If we focus on our two candidate selection methods, LDA_m (CS_m) and LDA_s (CS_s), the performance of LDA_m is higher than that of LDA_s. This shows that one of our assumptions, that a word can be a candidate keyword if it strongly belongs to at least one of topical clusters of a blog, is very effective. Comparing $Rich_m$ and LDA_m, and $Rich_s$ and LDA_s, it is very interesting that $Rich_m$ is better than LDA_m, and $Rich_s$ is better than LDA_m. If we recall that $Rich_m$ and $Rich_s$ choose good keywords from the candidates generated by LDA_m (CS_m) and LDA_s (CS_s), respectively, we can say that our richness scores considering the outside topical context are also very effective in extracting good and trendy keywords. If we compare $Rich_s$ and LDA_s, the performance of LDA_s is very low, but that of Richs is the second best. This also shows that our richness scores work well, even if the candidate set is roughly generated.

5. Conclusion

In this paper, a method was proposed to extract topic keywords from blogs, based on the web context. A new measure was proposed, *richness*, which indicated how well a blog reflected the web context. If a blog includes rich content on a topic word, the word can be considered as a keyword of the blog. Since the web includes various and trendy information on a topic, we could choose more popular and trendy keywords than other methods by measuring the richness of blogs, based on the web context. The trendy topic keywords would help the readers to understand the content of a blog more easily.

The results were compared with the traditional methods for keyword extraction, such as *TF-IDF, TextRank* and *HITS*, which are based on a statistical method, or a link analysis on the word-graph of texts. In the comparison, various evaluation measures were used; the hit counts, NDCG, trendiness and consistency. The proposed method showed superior results over the other methods.

Funding

This research was supported by Basic Science Research Program through the National Research Foundation of Korea funded by MEST (no. 2012-008062) and the IT R&D program of MKE/KEIT (KI001810041244, SmartTV 2.0 Software Platform).

Notes

1. http://www.techonorati.com
2. http://trends.google.com

References

[1] Wartena C, Brussee R and Slakhorst W. Keyword extraction using word co-occurrence. In: *Proceedings of the workshop on database and expert systems applications*, 2010, vol. 1, pp. 54–58.

[2] Liu S, Zhou MX, Pan S, et al. TIARA: Interactive, topic-based visual text summarization and analysis. *ACM Transactions on Intelligent Systems and Technology* 2012; 3: 791–800.

[3] Lu C, Park J and Hu X. User tags versus expert-assigned subject terms: A comparison of LibraryThing tags and Library of Congress Subject Headings. *Journal of Information Science* 2010; 36: 763–779.
[4] Sim H, Yoon T and Lee J-H. Creating related tag groups using co-occurrence frequency on blogsphere. In: *Proceedings of the international symposium on advanced intelligent systems*, 2009, vol. 1, pp. 187–190.
[5] Brooks CH and Montanez N. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In: *Proceedings of the 15th international conference on World Wide Web*, 2006, vol. 1, pp. 625–632.
[6] Thanadechteempat W and Fung CC. Automatic web content extraction for generating tag clouds form Thai web sites. In: *Proceedings of the IEEE 8th international conference on e-business engineering*, 2011, vol. 1, pp. 85–89.
[7] Razikin K, Goh DH, Chua AYK, et al. Social tags for resource discovery: A comparison between machine learning and user-centric approaches. *Journal of Information Science* 2011; 37: 391–404.
[8] Lee K-P, Kim H-G and Kim H-J. A social inverted index for social-tagging-based information retrieval. *Journal of Information Science* 2012; 38: 313–332.
[9] Sinclair J and Cardew-Hall M. The folksonomy tag cloud: When is it useful? *Journal of Information Science* 2007; 34: 15–29.
[10] Frank E, Paynter G, Witten I, et al. Domain-specific keyphrase extraction. In: *Proceedings of the sixteenth international joint conference on artificial intelligence*, 1999, vol. 1, pp. 668–673.
[11] Turney PD. Coherent keyphrase extraction via Web mining. In: *Proceedings of the 18th international joint conference on artificial intelligence*, 2003, vol. 1, pp. 434–439.
[12] Mihalcea R and Tarau P. TextRank: Bringing order into texts. In: *Proceedings of the conference on empirical methods on natural language processing*, 2004, vol. 4, pp. 404–411.
[13] Kleinberg J. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM* 1999; 46: 604–632.
[14] Chen B, Zhao Q, Sun B, et al. Predicting blogging behavior using temporal and social networks. In: *Proceedings of the 7th IEEE international conference on data mining*, 2007, vol. 1, pp. 439–444.
[15] Qamra A, Tseng B and Chang Y. Mining blog stories using community-based and temporal clustering. In: *Proceedings of the 15th conference on information and knowledge management*, 2006, vol. 1, pp. 58–67.
[16] Sekiguchi Y, Kawashima H, Okuda H, et al. Topic detection from blog documents using user's interests. In: *Proceedings of the 7th international conference on mobile data management*, 2006, vol. 1, pp. 108.
[17] Nallapati R and Cohen W. Link-PLSA-LDA: A new unsupervised model for topics and influence of blogs. In: *Proceedings of the international conference on Web and social media*, 2008, vol. 1, pp. 1019–1024.
[18] Ahmed A and Xing EP. Staying informed: Supervised and semi-supervised multi-view topical analysis of ideological perspective. In: *Proceedings of the conference on empirical methods in natural language processing*, 2010, vol. 1, pp. 1140–1150.
[19] Yano T, Cohen WW and Smith NA. Predicting response to political blog posts with topic models. In: *Proceedings of the annual conference of the North American Chapter of the Association for Computational Linguistics*, 2009, vol. 1, pp. 477–485.
[20] Karlgren J, Jonsson A, Boström H, et al. Automatic keyword extraction using domain knowledge. In: *Proceedings of the second international conference on computational linguistics and intelligent text processing*, 2001, vol. 1, pp. 472–482.
[21] Gazendam L, Wartena C and Brussee R. Thesaurus based term ranking for keyword extraction. In: *Proceedings of the 18th ACM conference on information and knowledge management*, 2009, vol. 1, pp. 235–244.
[22] Wimalasuriya DC and Dou D. Using multiple ontologies in information extraction. In: *Proceedings of the international conference on knowledge capture*, 2001, vol. 1, pp. 5–12.
[23] Blei DM, Ng AY and Jordan MI. Latent dirichlet allocation. *The Journal of Machine Learning Research* 2003; 3: 993–1022.
[24] Jarvelin K and Kekäläinen J. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 2002; 20: 422–446.
[25] Powers DMW. Applications and explanations of Zipf's law. In: *Proceedings of the joint conferences on new methods in language processing and computational natural language learning*, 1998, vol. 1, pp. 151–160.