An effective LDA-based time topic model to improve blog search performance

Article *in* Information Processing & Management · November 2017

DOI: 10.1016/j.ipm.2017.08.001

CITATIONS
13

READS
257

1 author:

Lin-Chih Chen
National Dong Hwa University
49 PUBLICATIONS **96** CITATIONS

SEE PROFILE

Information Processing and Management 53 (2017) 1299–1319

Contents lists available at ScienceDirect

Information Processing and Management

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

An effective LDA-based time topic model to improve blog search performance☆

Lin-Chih Chen

Department of Information Management, National Dong Hwa University, No. 1, Sec. 2, Da Hsueh Road, Shou-Feng, Hualien 97401, Taiwan

ARTICLE INFO

Article history:
Received 4 February 2017
Revised 17 July 2017
Accepted 14 August 2017

Keywords:
Blog search
Blog post
Time relationship
Natural language processing
Semantic analysis model

ABSTRACT

Blog search engines and general search engines automatically crawl web pages from the Internet and produce search results for users. One difference between the two is that blog search engines focus on posts and ignore the rest of the pages. Obviously, the pages indexed by the general search engine are always greater than the posts. This feature allows bloggers to focus only on the posts they are interested in, rather than other types of pages. The other difference is that posts involve more time-related issues compared to general pages. For the general pages, the general search engine often can only show the last update time. However, for the post, the blog search engine can display various possible times. For some often updated posts, the time factor can help bloggers find information more efficiently.

In this paper, we first use some well-known semantic analysis models to analyze the performance of the blog search. Next, we consider the time relationship between posts to further improve its performance. Finally, we provide some experiments to simulate various possible scenarios to confirm the effectiveness of this relationship. The contributions of this paper are twofold. One is that we build a high-performance system that considers the importance of blog topics at different times. The other is that we consider the time relationship between posts, which can rank the relevant blog topics based on the popularity of the posts.

1. Introduction

On the Internet, bloggers write relevant posts based on related events and blog service providers keep these posts. The key feature of posts is a group of people with similar interests discussing topics of mutual concern. Traditional page analysis focuses on text hyperlinks, while modern post analysis focuses on user comments (Fujimura et al., 2006; Jeong & Oh, 2012).

From the 1990s to 2015, the number of posts on the Internet grew from a few thousand to several million, and its growth rate was exponentially fast (Pingdom, 2015; Prayiush, 2015). In so many posts, how to use information technology to find useful information for the bloggers has become an important issue. Fortunately, blog search engines can give bloggers some useful posts based on the queries they enter. One difference between a general search engine and a blog search engine is that they index different objects (Mishne & Rijke, 2006; Thelwall & Hasler, 2007; Tsai, 2011). For all crawled pages, the general search engine indexes any page while the blog search engine indexes only the posts. The other difference is that they consider different time factors. General search engines usually only display the last update time on the page, but the

☆ The URL of the experimental system in this paper is at http://hlcs.sytes.net/ltr.

E-mail address: lcchen@mail.ndhu.edu.tw

http://dx.doi.org/10.1016/j.ipm.2017.08.001
0306-4573/

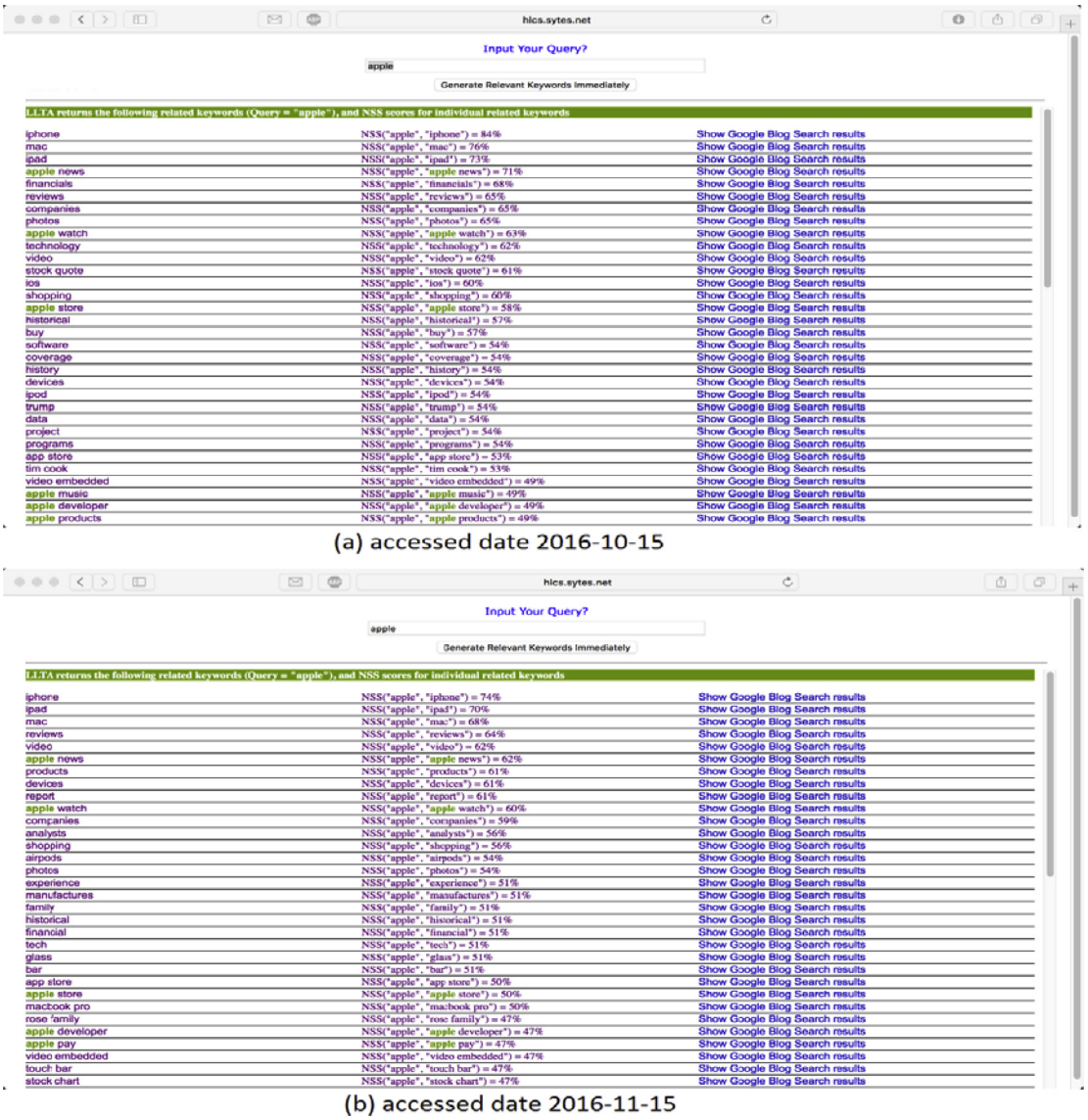


Fig. 1. A screenshot of our system when the query is "Apple".

blog search engine must display the various update times on the post. This difference is important because for the same topic, the meaning may be different when bloggers and subscribers discuss it at different times.

There are often latent topics or semantic relationships in posts (Geyer & Dugan, 2010; Husby & Barbosa, 2012). When bloggers express their personal opinions, the first question they think of is how to express the topic within the post. At the same time, subscribers can subscribe to certain posts with specific topics. In general, there may be a common discussion topic among similar posts, and the topic may also be relevant to the posting time of posts (Keikha, Gerani, & Crestani, 2011). Therefore, in blog research, we need a skillful way to deal effectively with the topic and time problems that exist in posts.

The latent semantic analysis model (later referred to as the "semantic model") can effectively identify the topic that exists between different documents (Blei, Ng, & Jordan, 2003; Hofmann, 2001; Landauer, Foltz, & Laham, 1998). In recent years, some well-known semantic models such as Latent Semantic Analysis (LSA), Probabilistic LSA (PLSA), and Latent Dirichlet Allocation (LDA) have emerged to find latent topics in documents. However, since these semantic models do not consider any time-related parameters (Yuan, Cong, Ma, Sun, & Thalmann, 2013), they cannot effectively solve the problem of the time factor in blog searches. This is important because, in an environment where documents are often updated, people often decide whether it is time-sensitive information based on the update time of the document.

In this paper, we complete a system to achieve the time relationship between posts, as shown in Fig. 1. For example, the "iphone" topic in the figure shows that some posts have iPhone discussion threads, and that these posts have similar update

Table 1
The discussion of blog search.

Compared to general search engines		
Difference	Benefit	
Index object	It can reduce the time users search for posts	
Time factor	It can help users choose time-sensitive posts	
Related research		
Work	Approach	Purpose
Jeong and Oh (2012)	Collecting some intelligent characteristics	A deeper search on a given topic
Keikha et al. (2013)	Deciding whether the sentences contain any product review terms	A suitable product review from the blog search
Wyner and Engers (2010)	Integrating comments, NLP, Ontology	Providing a rich online discussion forum
Zhu et al. (2011)	Analyzing the time-series issues and blog profile	Spam post detection
Takama et al. (2005)	The keyword map	Building an interactive blog search platform
Kim et al. (2015)	Ranking all crawled posts	Addressing unresolved information problems in the blogosphere

times. In addition, we show the similarity between the query and each topic, as shown by the NSS in the figure. We found that the second and third topics were “mac” and “ipad” at 2016-10-15 (Fig. 1(a)). However, at 2016-11-15 (Fig. 1(b)), the order of the two topics became “ipad” and “mac”. In addition, we found the topic “apple news” in the first case appears in the fourth topic, but in the second case is later. The first of these two cases focuses on the topics “mac” and “apple news” which are arranged in front. The main reason is that Apple released a new MacBook Pro in 2016-10-27, and before the release (the first case) there will be many related posts (or rumors) and news to discuss its possible hardware and software specifications. In the second case, Apple has released the product, so the discussion of related topics is not as good as the first case. The NSS results in the figure are also consistent with this statement. That is, in the second case, the NSS value of these topics is lower than the first case. In short, when the blog search considers the time factor, we can automatically cluster posts with different time characteristics to different topics based on the topic and time of the post.

The rest of this paper is organized as follows. Section 2 describes the literature relevant to this paper. Sections 3 and 4 discuss the research flow and related semantic models, respectively. Section 5 discusses and analyzes the relevant experimental results. Finally, in Section 6, we summarize this paper and present relevant future research work.

2. Literature review

In this section, we discuss three working models related to this study: the blog search, semantic model, LDA-based time topic models. To help the readers to read and compare the literature, we provide a table for each type.

2.1. Blog search related applications

Table 1 is the relevant description and comparison of the literature. In recent years, because the number of posts has increased rapidly, it is difficult for users to quickly find the relevant posts. An effective way to find relevant posts is to use the blog search engine to help users search. There are two differences between a general search engine and a blog search engine: index objects and time factors (Mishne & Rijke, 2006; Thelwall & Hasler, 2007). For the first difference, the general search engine indexes all of the crawled pages, while the blog search engine indexes only the posts. Obviously, the total number of pages is higher than the number of posts. Therefore, the blog search engine only needs to focus on the post rather than the entire page, which can significantly reduce the user’s search time. For the second difference, the general search engine only needs to display the last update time on the page, while the blog search engine needs to display various update times on the post. For some often updated posts, the time factor can effectively help users to select those time-sensitive posts. Today, there are some well-known blog search engines on the Internet, such as Google Blog Search,[1] Technorati,[2] and Regator.[3]

Several researchers have successfully used different blog search techniques to improve the performance of their blog search. Jeong and Oh (2012) proposed a blog search architecture that can perform a deeper search on a given topic. By collecting some intelligent characteristics in the posts, it can further change the original topic. Keikha, Crestani and Carman (2013) proposed a method to find a suitable product review from the blog search. The method needs to decide whether the sentences in the posts contain any product review terms or not. The deciding method involves the following two steps. First, it needs to build the terms’ features for each product. Then, it sends different features to a blog search engine and

[1] https://www.google.com/search?tbmnws&tbs=nrt:b.
[2] http://www.technorati.com/.
[3] http://regator.com/.

Table 2
Comparison of different semantic models.

Idea	Model		
	LSA	PLSA	LDA
Approach	SVD & Dimensionality reduction	EM algorithm	Dirichlet probability distribution & Gibbs sampling
Advantage	Synonymy	Synonymy & Polysemy	Unsupervised generative model
Disadvantage	Polysemy	Computing time	Not suitable for a small amount data or normal distribution
Problem solving	Document summarization, Answer assessment, Music analysis, Ranking algorithm, Plagiarism detection	Sound event detection, Location prediction, Sports video semantic event detection, Scene image recognition	Recommended multimedia tags, Bug reports, Image comment, Bug Localization

collects all posts to sort out all of the product review terms. Wyner and Engers (2010) proposed a blog search architecture based on a blog search of e-government. Based on the architecture, they provided a rich online discussion forum that can show the pros and cons of some methods and clusters all opinions into public opinion. They used the following methods to build the discussion forum: integrating comments, Natural Language Processing (NLP), ontology. Zhu, Sun and Choi (2011) presented spam post detection technology based on the search results returned from the blog search engine. The technology analyzes the search results based on some time-series issues, and builds a blog profile that contains the top search results. Finally, they used the profile to store all of the spam posts to help users to filter out unimportant spam information. Takama, Kajinami and Matsumura (2005) used the keyword map to build an interactive blog search platform. Based on the idea of multiple topics and many conversations, the platform can tackle different topics based on the keyword map. Kim et al. (2015) proposed an algorithm that efficiently performs a ranking on the posts to address unresolved information problems in the blogosphere.

2.2. A comparison of semantic models

Currently, the most famous semantic models are LSA (Landauer et al., 1998), PLSA (Hofmann, 2001), and LDA (Blei et al., 2003). In this subsection, we compare and analyze these semantic models based on the ideas of approach, advantages and disadvantages, and problem solving, as shown in Table 2. In Section 4, we will describe the details of these semantic models.

To separate the topic matrix from the original term-document matrix, LSA first uses Singular Value Decomposition (SVD) to decompose the term-document matrix into three separate matrices. To filter out the noisy topics from the topic matrix, it then uses a dimensionality reduction technique to reduce the topic matrix to K largest dimensions. The advantage of LSA is that it deals with the problem of synonyms of terms by dimensionality reduction (Dumais, 2004). In contrast, the disadvantage of LSA is that it cannot handle the problem of polysemy for terms because SVD is only a one-to-one mapping from a specific term to a specific document (Landauer, McNamara, Dennis, & Kintsch, 2013).

There are many researchers who used LSA to address different Information Retrieval (IR) problems. For the problem of document summarization in many user-generated documents, some researchers (Ozsoy, Alpaslan, & Cicekli, 2011; Yeh, Keb, Yang, & Meng, 2005) used LSA to sort all documents to achieve the purpose of document summarization. For the assessment of short free text answers, there have been some researchers (Klein, Kyrilov, & Tokman, 2011; Lintean, Moldovan, Rus, & McNamara, 2010) who tried to use LSA to assess automatically whether the answer is correct. For music analysis, some researchers (Kuo, Shan, & Lee, 2013; Logan, Kositsky, & Moreno, 2004) applied LSA to address the music noise problem of the background music. For the search engine ranking issues, Luh, Yang and Huang (2016) used LSA and a genetic algorithm to simulate Google's ranking algorithm. For the problem of plagiarism detection in source code, Cosma and Joy (2012) tried to use LSA to decide whether a piece of source code has been plagiarized.

PLSA uses an iterative Expectation Maximization (EM) algorithm to calculate the probability of the cells in the term-document matrix. The algorithm first estimates the expected value of the hidden topic based on the probability of observed parameters. Then, it updates the probability of observed parameters by the objective function maximization. Compared with LSA, the advantage of PLSA is that it can further deal with the problem of polysemy for terms because the EM algorithm is a statistical estimation method that can simultaneously estimate multiple parameters (Fernandez-Beltran & Pla, 2015; Siddiqui, Mishra, & Verma, 2015). However, the disadvantage of PLSA is that its computational time is huge because the EM algorithm is a time-consuming method (Brahmane & Amune, 2014; Hsieh et al., 2015; Li, Li, & Li, 2013).

There have been many researchers in recent years who have successfully applied PLSA to addressing different IR problems. Mesaros, Heittola and Klapuri (2011) used a two-stage approach to detect human motion sound events. In the first stage, they used PLSA to find possible associations between events. In the second stage, they used PLSA again to update the probability of events based on the historical distribution of events. McInerney, Rogers and Jennings (2012) used PLSA to improve the location prediction services that try to predict the mobility patterns for some new users. For sports video sum-

Table 3
A comparison of LDA-based time topic models.

Work	Model	Practice	Focus
Wang and McCallum (2006)	TOT	Word co-occurrence mode	• Particular topic occurred • Beta distribution describes the time parameters
Bolelli et al. (2009)	S-ATM	Specific time segment	• Analyze trends in specific topics • Time segment is a discrete event
Zhao et al. (2011)	Twitter-LDA	Automatically collect topics, semi-automatically group topics, manual classify topics	• Semi-automatic or manual classification topics • Topics are defined by NYT
Wang et al. (2012)	TM-LDA	Training topic transition parameters	• Long time to train the parameters • Parameters are the overall trend for all documents.

marization and retrieval, Xu, Zhang, Zhu, Rui, Lu and Huang (2008) first analyzed a text drawn from the broadcast video and then used PLSA to detect all of the possible sport events. Ji, Jing, Wang and Su (2012) used scattered space block information and PLSA to classify image scene recognition into computer vision.

LDA first uses the Dirichlet probability distribution to set up the latent probability of documents. Then, it uses the Gibbs sampling algorithm to estimate the final probability values of topics and terms for a given document. The advantage of LDA is that it can easily find all matching terms for each topic because it is an unsupervised generative model without any prior information about topics and terms (Blei et al., 2003). In contrast, the disadvantage of LDA is that it is not suitable for the environment of a small amount data or normal distribution (Liu, Zhang, Chang, & Sun, 2011; Wang & Blei, 2013).

To help the user to search multimedia content, Krestel, Fankhauser and Nejdl (2009) used LDA to recommend relevant multimedia tags to users. To automatically classify many bug reports, Somasundaram and Murphy (2012) used LDA to help users simplify the debugging process in code compilation. For performing an efficient image comment, Liénou, Maître and Datcu (2010) first searched all patterns between different image frames. They then applied LDA to comment automatically on the image according to the pattern distribution. To perform the bug localization on a given source code, Lukins, Kraft and Etzkorn (2008) first built an LDA model to extract semantic information from the code. Then, they created another LDA model to localize possible bugs based on known semantic information.

2.3. LDA-based time topic models

LDA has proven to be an effective unsupervised learning method for finding topics that exist between documents. Since LDA does not take any time parameters, it cannot effectively identify possible changes in the topic of the document (Wang & McCallum, 2006). Some research has been based on LDA and different time signals to develop various LDA-based time topic models. Table 3 is a comparison of LDA-based time topic models. Wang and McCallum (2006) proposed a TOT model to extract the topic changes from the time offsets in all documents. The model uses the word co-occurrence mode to simulate the relevant time parameters. The model focuses on the following two points. One is that its time parameter is based on the time that a particular topic occurred. The other is that it uses beta distribution to describe the time parameter. Since the beta distribution is the conjugate probability distribution of the binomial distribution, it represents that its time parameter is chosen in two states. Bolelli, Ertekin and Giles (2009) proposed an S-ATM model to explore the trends in related research topics in academic literature. The model uses a specific time segment as a time parameter to estimate the trend of the change in the research topic caused by a time shift. The model focuses on the following two points. One is that it analyzes the trend of topic changes for some specific research topics. Since the coverage of general research topics is more comprehensive, these topics can be predefined. The other is that the time segment it uses is described as a discrete event. Zhao et al. (2011) proposed a Twitter-LDA model to classify the topics on Twitter based on the chronological order in the New York Times (NYT) news articles. They first use Twitter-LDA to automatically collect possible discussion topics on Twitter. Then they use the topics in the NYT to semi-automatically group the relevant topics on Twitter into different topic categories. Finally, they manually classify all topics into different topic types. The model focuses on the following two points. One is that it uses a semi-automatic or manual way to group or classify topics. The other is that its topic categories are based on the topics defined by the NYT. Wang, Agichtein and Benzi (2012) proposed a TM-LDA model to predict future topic trends for new documents. The model uses historical temporally-sequenced documents to train topic transition parameters and predicts the future topic distribution of new documents by parameters. The model focuses on the following two points. One is that it produces relative training parameters by training many sample documents, so it takes a

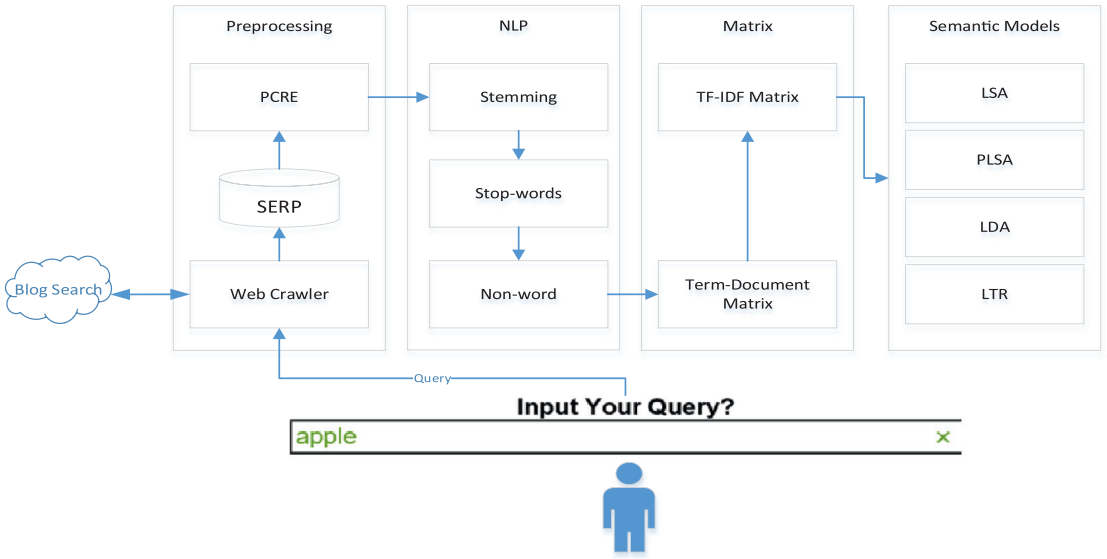


Fig. 2. The research flow of this study.

long time to train the parameters correctly. The other is that its training parameters are trained through all the documents, which means that its parameters are the overall trend of all documents.

3. Research flow

Fig. 2 is the research flow of this study, which contains Preprocessing, NLP, Matrix and Semantic Models steps. In the first step, we develop an intelligent crawler to crawl the Search Engine Results Page (SERP) returned from a blog search. In the second step, we apply some NLP techniques to convert unstructured HTML documents into structured documents. In the third step, we convert all the structured documents into a term-document matrix, which is the input of different semantic models. In the last step, we use different semantic models to find the topic relationships between documents.

3.1. Preprocessing step

The purpose of this step is to crawl the SERP returned by the blog search as a source of data for this study. In order to quickly access the data source, in this study, we develop a multi-threading technique to crawl all the returned SERPs at the same time. In theory, although we do not consider network bandwidth and network latency issues, through our crawler technology, we can attain a crawl time for multiple SERPs that is equal to one SERP.

A SERP is the HTML document returned by the blog search for a given query. Since the HTML document is an unstructured document type (Chen, 2012; Shi, Quan, & Li, 2013), we need a way to find some important analysis items from the document, including the title, URL and a brief description. In this study, we use the Perl Compatible Regular Expressions (PCRE) library (Hazel, 2017), which is a well-known pattern matching method, to help us get relevant analysis items from an HTML document.

3.2. NLP step

The purpose of this step is to use some NLP techniques, including Stemming, Stop-words, and Non-words,[4] to transform the analysis items into a structured document. For the stemming technique, since each word in the analysis items may have some different forms, we need to aggregate the same word with different forms into its root form. In this study, we use the Porter stemming algorithm (Porter & Boulton, 2017) to perform this aggregation work. It is a well-known stemming algorithm that converts words with different parts of speech (noun, verb, adjective, adverb) into their root form. For the stop-words technique, since stop words are meaningless and can affect information retrieval performance, we must remove all possible stop words. In this study, we use the 421 stop words suggested by Fox (1989) as a basis and expand the common stop words on the Internet as our stop words dictionary. In fact, the main sources of our expanded stop words include Google (2014) and MySQL (2017), and we have removed the recurring stop words from these sources to get the final stop dictionary. At present, our stop word dictionary has 1085 stop words. For the same reason, in the Non-word technique, all non-word characters must also be removed.

[4] According to the PCRE definition, a non-word character is any character that is not alphanumeric or an underscore.

3.3. Matrix step

Since different semantic models take the term-document matrix as the source, the purpose of this step is to convert the structured document into a matrix form. In this study, we use the Term Frequency Inverse Document Frequency (TFIDF) statistical method to perform this conversion work, as shown in the following equation:

$$TFIDF_{i,j} = m_{i,j}/m_i \times \log(M/df_j) \tag{1}$$

where $TFIDF_{i,j}$ is the weight of the term t_j in the document d_i in the matrix, $m_{i,j}$ is the number of occurrences of term t_j in document d_i, m_i is the total number of terms in document d_i, M is the number of documents in the corpus, and df_j is the number of documents where the term t_j occurs.

The spirit of TFIDF is that if the frequency of a specific term t_j in a specific document d_i is high and other documents appear less, it should be given a higher weight to the term t_j in document d_i.

4. Semantic models

In this section, we first briefly explain the concepts of LSA, PLSA, and LDA, and describe their associated statistical models. Next, we present our Latent Time Relationship (LTR) model and compare it with other models.

4.1. LSA

LSA is a semantic model based on SVD. It first uses SVD to capture the topic space from the term-document matrix. Then, it applies the dimensionality reduction technique to remove the noise in the original topic space. The following equation represents the details of the SVD, where A is the term-document matrix, S is the topic space, U is the term space corresponding to S, and V is the document space corresponding to S.

$$A = USV^T \tag{2}$$

By using SVD, the term-document matrix is first broken down into three matrices, where U holds the term information, S is a diagonal matrix which contains the topic space of A, and V holds the document information. Then, it uses the dimensionality reduction technique to remove the noise topic in S. That is, it preserves the K largest topics in S and the rest as noise topics in order to obtain the noise-free topic space S'. Finally, the three matrices U, S', V^T are re-multiplied to produce a noise-free term-document matrix.

4.2. PLSA

PLSA uses the Expectation Maximization (EM) algorithm and the Maximum Likelihood Estimation (MLE) method to estimate the occurrence probability $P(d_i,t_j)$ of term t_j in document d_i in the term-document matrix.

The PLSA-related parameters are described below. A specific topic z_k is selected when given a specific document d_i and a specific term t_j is chosen when given a known z_k. In PLSA, the parameter (d_i,t_j) represents the term-document matrix, where $d_i \in \{d_1,\ldots,d_M\}$ is a specific document, and $t_j \in \{t_1,\ldots,t_N\}$ is a specific term. The parameter $z_k \in \{z_1,\ldots,z_K\}$ is an unobserved topic variable. The following probability parameters are used to estimate $P(d_i,t_j)$.

- $P(d_i)$ is the probability that document d_i occurs.
- $P(t_j|z_k)$ is the conditional probability that term t_j occurs when a given topic z_k occurs.
- $P(z_k|d_i)$ is the conditional probability that topic z_k occurs when a given document d_i occurs.

Based on Bayes' theorem, the following equation is used to estimate $P(d_i,t_j)$, where $P(t_j|z_k)$, $P(z_k)$, and $P(d_i|z_k)$ are its target parameters.

$$P(d_i, t_j) = P(d_i)\sum_{k=1}^{K} P(t_j|z_k)P(z_k|d_i) = \sum_{k=1}^{K} P(t_j|z_k)P(z_k)P(d_i|z_k) \tag{3}$$

PLSA uses the EM algorithm and the MLE method to estimate the relevant target parameters. The relevant estimation procedure is as follows. First, it uses the following equation as the objective function and maximizes the function by the MLE method, where n is the iteration number, $w(d_i,t_j)$ is the weight of term t_j in document d_i.

$$L_n(d_i, t_j) = \sum_{j=1}^{N}\sum_{i=1}^{M} w(d_i, t_j)\log P(d_i, t_j) \tag{4}$$

Next, it uses the EM algorithm to maximize the target parameters. The algorithm repeats the expectation E and maximization M steps until the termination condition is reached. In step E, the algorithm uses the observed target parameters to estimate the probability of an unobserved topic, as shown in the following equation.

$$P(z_k|d_i, t_j) = \frac{P(d_i|z_k)P(z_k)P(t_j|z_k)}{\sum_{k=1}^{K} P(d_i|z_k)P(z_k)P(t_j|z_k)} \tag{5}$$

In step M, the algorithm uses the result of step E to maximize the target parameters, as shown in the following equations.

$$P(t_j|z_k) = \frac{\sum_{j=1}^{N} w(d_i, t_j)P(z_k|d_i, t_j)}{\sum_{j=1}^{N}\sum_{i=1}^{M} w(d_i, t_j)P(z_k|d_i, t_j)} \tag{6}$$

$$P(z_k) = \frac{\sum_{j=1}^{N}\sum_{i=1}^{M} w(d_i, t_j)P(z_k|d_i, t_j)}{\sum_{j=1}^{N}\sum_{i=1}^{M} w(d_i, t_j)} \tag{7}$$

$$P(d_i|z_k) = \frac{\sum_{i=1}^{M} w(d_i, t_j)P(z_k|d_i, t_j)}{\sum_{j=1}^{N}\sum_{i=1}^{M} w(d_i, t_j)P(z_k|d_i, t_j)} \tag{8}$$

4.3. LDA

LDA first uses the Dirichlet probability distribution to set up the latent probability of documents. Then, it uses the Gibbs sampling algorithm to estimate the final probability values of topics and terms for a given document. The LDA-related parameters are described below.

- α is the parameter of the Dirichlet distribution prior to the topic-document distributions.
- β is the parameter of the Dirichlet distribution prior to the term-topic distributions.
- θ_d is the topic distribution of document d, which corresponds to the conjugate distribution of parameter α in the Dirichlet distribution.
- φ_z is the term distribution of topic z, which corresponds to the conjugate distribution of parameter β in the Dirichlet distribution.
- $z_{d,t}$ is the topic of term $t$in document d, which corresponds to the conjugate distribution of parameter θ_d in the multinomial distribution.
- t_d is the specific term in document d, which corresponds to the conjugate distribution of parameter φ_z in the multinomial distribution.
- K is the number of topics, N is the number of terms.

LDA uses the Gibbs sampling algorithm and the MLE method to estimate the final probability values of α and β for a given document. The relevant estimation procedure is as follows. It first estimates the conditional probability of a K dimensional Dirichlet variable θ given the parameter α ($\theta_k \geq 0$ and $\alpha_k \geq 0$), as shown in the following equation; where α is a K-vector with elements α_k, $\Gamma(x)$ is the Gamma function.

$$P(\theta|\alpha) = \frac{\Gamma\left(\sum_{k=1}^{K}\alpha_k\right)}{\prod_{k=1}^{K}\Gamma(\alpha_k)}\theta_1^{\alpha_1-1}\ldots\theta_k^{\alpha_k-1}\ldots\theta_K^{\alpha_K-1} \tag{9}$$

It next estimates the joint distribution probability of all observed and unobserved parameters given the partial observed parameters, α and β, as shown in the following equation.

$$P(t_d, z_{d,t}, \theta_d, \varphi_z|\alpha, \beta) = \prod_{t=1}^{N} P(\theta_d|\alpha)P(z_{d,t}|\theta_d)P(t_d|\varphi_z)P(\varphi_z|\beta) \tag{10}$$

It then estimates the MLE of the term distribution of a document by summing over θ_d and φ_z, as shown in the following equation.

$$P(t_d|\alpha, \beta) = \sum_{\theta_d}\sum_{\varphi_z} P(t_d, z_{d,t}, \theta_d, \varphi_z|\alpha, \beta) \tag{11}$$

It finally uses the Gibbs sampling algorithm (Speh, Muhic, & Rupnik, 2013) to estimate the topics from the collected documents as well as the topic-document α and term-topic β probabilities. Compared with PLSA, it has the advantage that it can significantly reduce the computation time because it uses the Gibbs sampling algorithm rather than the EM algorithm to estimate the relevant parameters (Speh et al., 2013).

Table 4
LTR related considerations and selection reasons.

Considerations	Our selection	Reason
Focus	Time factor	The topic of blog search is closely related to the time factor
Underlying semantic model	LDA	1. The relationships between topics and terms for a given document 2. The computation time is relatively fast
Graphics model	Aspect model	A graphic model of more simple explanation
Parameters	γ	The topic-time distributions
	v_{tr}	The topic distribution of time relationship *tr*

4.4. LTR

LSA, PLSA and LDA can effectively identify the topic relationship between different documents. However, since these semantic models do not consider any time-related parameters (Yuan et al., 2013), they cannot effectively solve the problem of the time factor in blog searches. Table 4 shows the LTR model considerations and selection reasons. In general, the emergence of a hot topic is usually associated with the post update time because these posts are often in a similar timeframe for discussing the same topic. Therefore, compared to other semantic models, LTR focuses on how to properly set the time factor for a blog search. There are two reasons LTR chooses LDA as the underlying semantic model. The first reason is that LDA focuses on finding relationships between topics and terms for a given document. The second reason is that the LDA uses the Gibbs sampling algorithm to estimate the relevant parameters, so its computational time is much faster than PLSA. To simplify the complexity of LTR, we use a graphic model called the aspect model (Hofmann, 2004) to explain the relevant parameters and estimation process, as shown in Fig. 3.

In the LTR, we add the time factor γ and produce a binary link variable v_{tr}. The LTR-related parameters are described below.

- γ is the parameter of the Dirichlet distribution prior to the topic-time distributions.
- v_{tr} is the topic distribution of time relationship *tr*, which corresponds to the conjugate distribution of parameter γ in the Dirichlet distribution.

LTR also uses the Gibbs sampling algorithm and the MLE method to estimate the final probability values of α, β, and γ for a given document. The relevant estimation procedure is as follows.

It first estimates the joint distribution probability of all observed and unobserved parameters given the partial observed parameters, α, β, and γ, as shown in the following equation.

$$P\left(t_d, z_{d,t}, \theta_d, \varphi_z, v_{tr}|\alpha, \beta, \gamma\right) = \prod_{t=1}^{N} P(\theta_d|\alpha)P\left(z_{d,t}|\theta_d\right)P(t_d|\varphi_z)P(\varphi_z|\beta)P\left(z_{d,t}|v_{tr}\right)P(v_{tr}|\gamma) \tag{12}$$

It then estimates the MLE of the term distribution of a document by summing up θ_d, φ_z, and v_{tr}, as shown in the following equation.

$$P(t_d|\alpha, \beta, \gamma) = \sum_{\theta_d}\sum_{\varphi_z}\sum_{v_{tr}} P(t_d, z_{d,t}, \theta_d, \varphi_z, v_{tr}|\alpha, \beta, \gamma) \tag{13}$$

It finally uses the Gibbs sampling algorithm to estimate the topics from the collected documents as well as the topic-document α, term-topic β, and topic-time γ probabilities.

4.5. LTR discussion and comparison

Table 5 is another comparison of different semantic models. LSA uses SVD and dimensionality reduction to filter out possible noise topics in the original data. Through dimensionality reduction, it can effectively deal with the problem of synonyms for terms. PLSA uses the EM algorithm and MLE method to estimate the relevant parameters. It can further deal with the problem of polysemy for terms because the EM algorithm can simultaneously estimate different parameters. LDA and LTR use the Gibbs sampling algorithm and MLE method to estimate the relevant parameters. The main difference between the two models is that they consider different perspectives. LDA focuses on finding topic relationships between documents, because its two main parameters, α (topic-document) and β (term-topic), are based on the topic. LTR focuses on finding the time factor between topics because its main parameter γ (topic-time) is based on time.

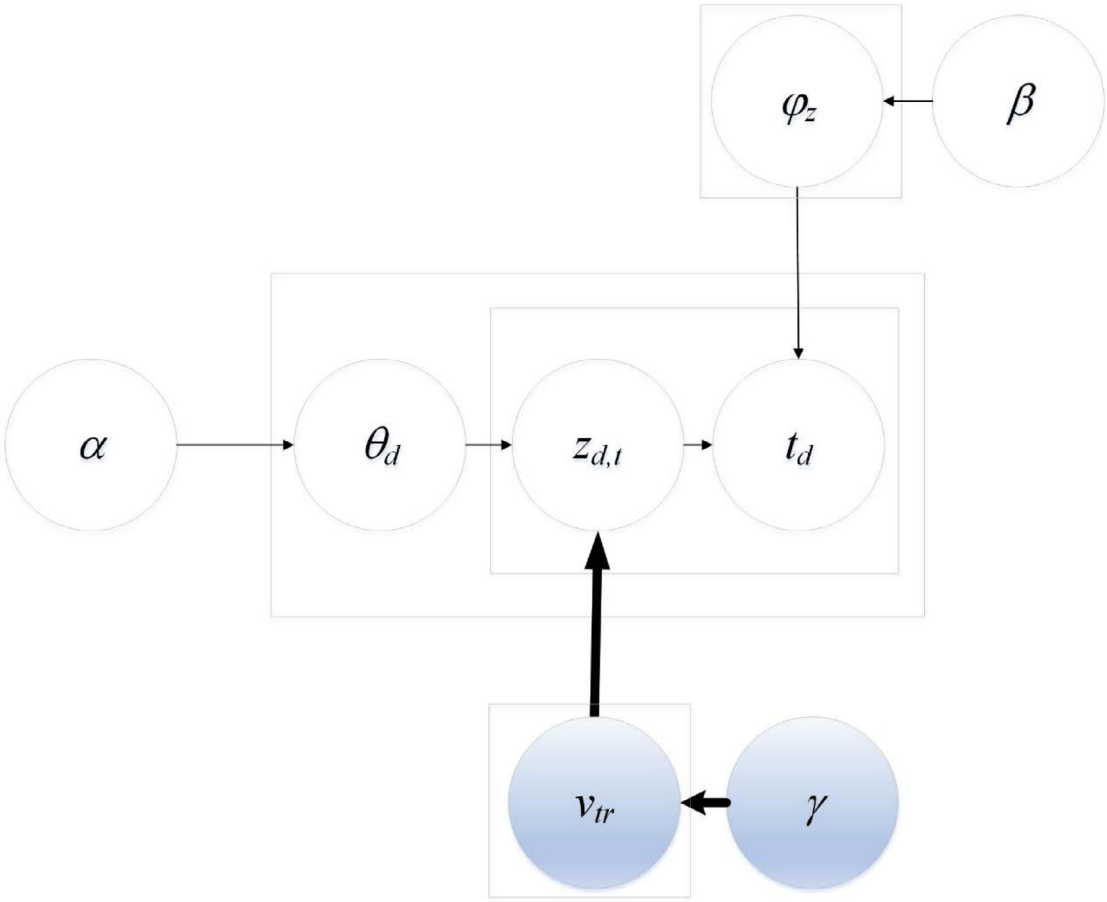


Fig. 3. LTR graphics model.

Table 5
Another comparison of different semantic models.

	Parameters (Equation)		
Model	Estimation method	Speed	Focus
LSA	SVD & Dimensionality Reduction U (term space), S (topic space), V (document space) (Eq. (2))	Fastest	Synonyms
PLSA	EM & MLE $P(t_j\|z_k)$ (U's probabilistic form), $P(z_k)$ (S's probabilistic form), $P(d_i\|z_k)$ (V's probabilistic form) (Eq. (3))	Slowest	Polysemy
LDA	Gibbs & MLE α (topic-document distribution), β (term-topic distribution) Eqs. (10) & ((11))	Fast	Topic
LTR	Gibbs & MLE α (topic-document distribution), β (term-topic distribution), γ (topic-time distribution) Eqs. (12) & ((13))	Fast	Time factor

Next, we discuss the differences in computational speed between different semantic models. LSA is the fastest because it does not use any iterative method to estimate the relevant parameters. On the contrary, the speed of PLSA is the slowest because the computation time of the EM algorithm exponentially increases with the number of terms and documents (Hofmann, 2004). LDA and LTR are significantly faster than PLSA because they use the Gibbs sampling algorithm rather than the EM algorithm to estimate the relevant parameters. In the next section, we will compare the performance and cost between the different models.

The advantage of the LTR is that it can cluster similar documents with similar update times into a cluster through the time factor. Its spirit is that when documents have similar update times and discussion topics, the meaning of their discussions is similar. However, for documents with similar discussion topics but different update times, LTR can clearly distinguish that the meaning of their discussion is different. For example, since 2007, Apple have released at least a new iPhone ev-

ery year. By LTR processing, we can clearly distinguish between the iPhone 6s and iPhone 7, because these two types have different release dates.

5. Experimental analysis and comparison

In this section, we first discuss the experimental environment associated with this study. Then, we analyze and compare the performance and cost differences between different semantic models. Finally, we compare the performance of different LDA-based time topic models.

5.1. Experimental environment

In this experiment, we discuss the experimental environment which consists of the following three parts: test data set, training corpus and performance indicator.

- Test Data Set
  The test data set used in this study is a set of 1000 test queries, which are real queries for people searching on Google Blog Search from 2013/3/13 to 2016/11/8. Since the number of test data sets is large, we encourage interested readers to refer to it at http://hlcs.sytes.net/ltr/randomly_1000.pdf. For each test query, the Google Blog Search returns about 350 to 400 posts. Therefore, for each query, we select the first 350 posts as experimental data sources.
- Training Corpus
  Because the test data set and performance indicator are based on the Google Blog Search, we use the cached results from the Google Blog Search as our training corpus so that we can evaluate the performance difference between the different models. Google has now integrated the Google Blog Search into Google News Search. To access this corpus, we must add Google's defined news and blog parameters to the Google search. That is, we calculate the number of hits that are returned when we added the news ("tbm=nws") and blog ("tbs=nrt:b") parameters to the test query on the Google search.
- Performance Indicator
  There are two reasons for this study to use a computer simulation rather than a manual evaluation. On the one hand, because the number of test queries is large, it is not possible to perform a manual evaluation for a limited time. On the other hand, we are able to assess the performance difference between different semantic models in a more objective way.

Since the results of this study are topics, we must find a suitable performance indicator to calculate the semantic similarity between the test query and the topic. To allow our simulation program to accurately define the semantic similarity between any two topics, we use an MSR (Measure of Semantic Relatedness) statistical technique (Lindsey, Veksler, Grintsvayg, & Gray, 2007) called NSS (Normalized Similarity Score) (Batra & Bawa, 2010; Chandramohan, Khapre, & Ashokkumar, 2011; Cilibrasi & Vitányi, 2007; Liu, Chen, & Ho, 2015; Veksler, Govostes, & Gray, 2008) to help the program automatically discover similar information that exists between topics in the corpus. Other MSRs (Patwardhan, Banerjee, & Pedersen, 2003), such as (Leacock & Chodorow, 1998; Resnik, 1995; Jiang & Conrath, 1997; Lin, 1998; Hirst & St-Onge, 1998), are computationally time-consuming and can only handle a corpus with less data. As a result, these MSRs cannot handle large web-based corpora (Cilibrasi & Vitányi, 2007). The formula for NSS is shown in the following equation:

$$NSS(x,y) = 1 - \frac{\max(logf(x), logf(y)) - logf(x \text{ and } y)}{logM - \min(logf(x), logf(y))} \tag{14}$$

where M is the total number of pages in the corpus, x is the test query, y is the topic name, $f(Q)$ is the number of hits returned by the query $Q \in \{x, y, x \text{ and } y\}$ in the Google search engine.

When the NSS value is bigger, the semantic similarity of two topics is higher. For example, in Fig. 1(a), we found the NSS values for the first three topics were 84% (y = "iphone"), 76% (y = "mac"), 73% (y = "ipad") where x = "apple". Interested readers can use the following URL to verify our simulation program: http://hlcs.sytes.net/ltr/experiment_nss.php.

5.2. Performance evaluation of different semantic models

Among the different models, the number of topics will affect the performance of the model. In general, a suitable number of topics can achieve the following two purposes: (a) the meaning within the topic is closer, and (b) the meaning between the topics is farther apart. When the number of topics is more, the difference between topics is lower while conversely, when the number of topics is less, the meaning within the topic is more dispersed. Therefore, how to set a suitable number of topics has become one of the important issues in this study. According to the suggestion of Hofmann, Schölkopf and Smola (2008), we set the number of topics from 5 to 50 to evaluate the performance of different models.

First, we perform an experiment with a number of topics equal to 5. Fig. 4(a) is the average NSS value between the test query and all its related topics when the number of test queries equals 1000. In the figure, the x-axis represents the number of test queries and the y-axis represents the average NSS value. For convenience, each point in the figure is the average NSS value for 100 test queries. According to the results in the figure, we found the performance of LSA is the lowest because it

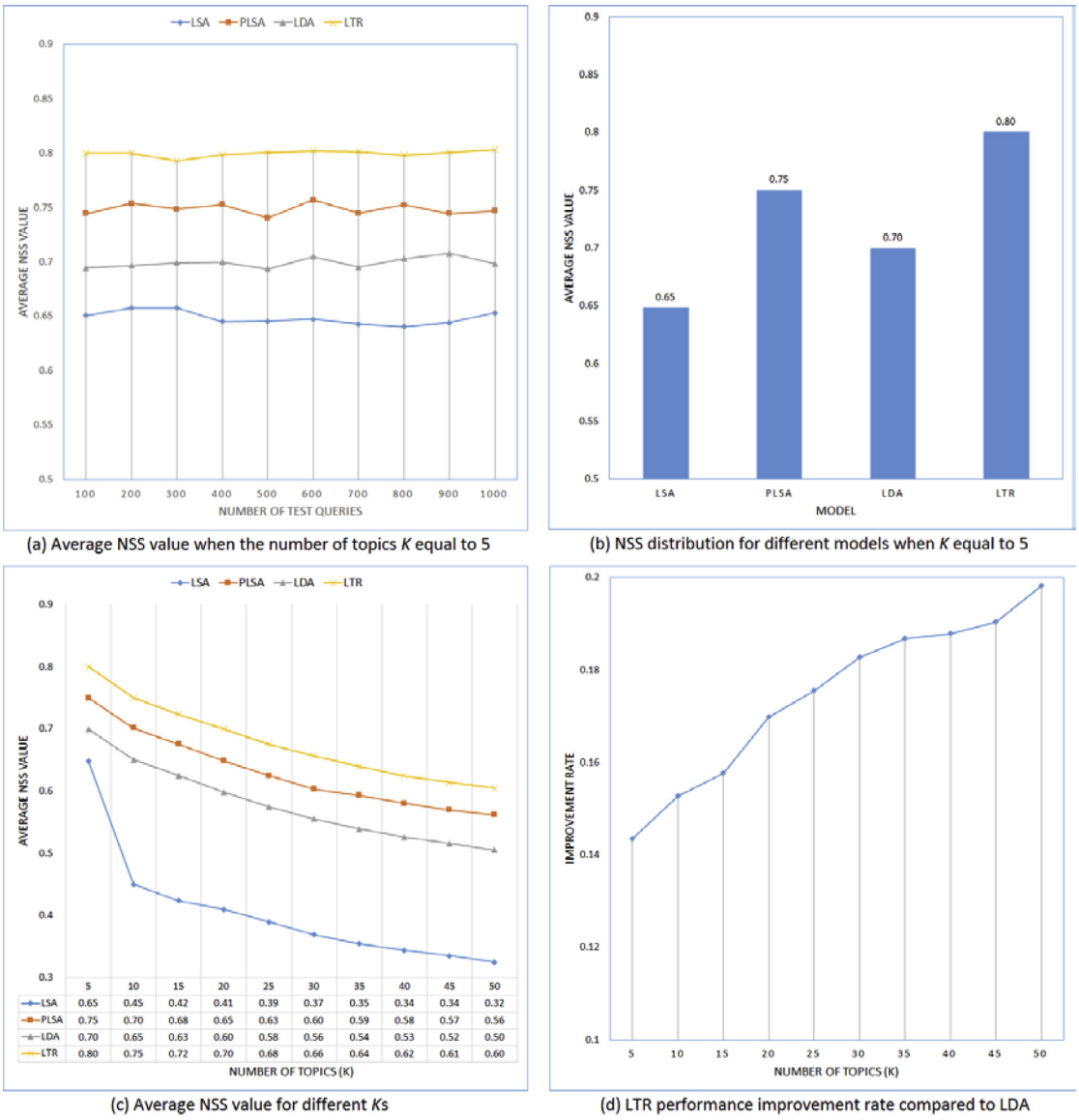


Fig. 4. Average NSS values for different models.

can only find the query-related synonymous topics (Dumais, 2004; Landauer et al., 2013). Compared with other models, the number of these synonymous topics is not only small, but also focused on a few topics. PLSA, LDA, and LTR can further find the query-related polysemous topics, so they perform better than LSA. According to the experimental results, we found that the PLSA is superior to LDA in dealing with polysemy topic. However, according to the experimental results in Section 5.3, the computation time of PLSA is significantly longer than other models. These blog search queries contain many queries that have the same topic but differ in their meaning because of the time difference. For example, the topic “black friday” with the numbers 97 to 99 and the topic “iphone” with the numbers 417 to 426. For often updated blog topics, increasing the time factor can effectively distinguish the same topic at different times to show different meanings. LTR is superior to other models because it increases the time factor that is often considered in the blog search. To ease the comparison below, we averaged all NSS values for different models to build the NSS distribution for LSA, PLSA, and LDA, as shown in Fig. 4(b).

Next, we extend the experiment to a different number of topics K, where $K \in \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$. Fig. 4(c) is a comparison of average NSS values when using different Ks. In the figure, the x-axis represents the number of topics K. According to the results in the figure, the average NSS value decreases as the number of topics K increases. When K is greater than 5, the average NSS value of LSA has a tendency to decrease obviously, but the trend of other models is not obvious. This means that LSA is not suitable for handling multiple topics at the same time. Although the average NSS value of PLSA is superior to LDA, it needs a long computation time to reach this solution, as described later. This means that PLSA

Table 6
ANOVA analysis results for different models.

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	427.984	3	142.661	25,671.936	.000
Within Groups	222.261	39,996	.006		
Total	650.245	39,999			

Table 7
The LSD analysis results for different models.

(I) Models	(J) Models	Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
					Lower Bound	Upper Bound
LSA	PLSA	−0.22583(*)	0.00105	0.00	−0.22790	−0.22377
	LDA	−0.17355(*)	0.00105	0.00	−0.17562	−0.17149
	LTR	−0.27383(*)	0.00105	0.00	−0.27589	−0.27176
PLSA	LSA	0.22583(*)	0.00105	0.00	0.22377	0.22790
	LDA	0.05228(*)	0.00105	0.00	0.05021	0.05435
	LTR	−0.04799(*)	0.00105	0.00	−0.05006	−0.04593
LDA	LSA	0.17355(*)	0.00105	0.00	0.17149	0.17562
	PLSA	−0.05228(*)	0.00105	0.00	−0.05435	−0.05021
	LTR	−0.10027(*)	0.00105	0.00	−0.10234	−0.09821
LTR	LSA	0.27383(*)	0.00105	0.00	0.27176	0.27589
	PLSA	0.04799(*)	0.00105	0.00	0.04593	0.05006
	LDA	0.10027(*)	0.00105	0.00	0.09821	0.10234

*The mean difference is significant at the .05 level

Table 8
The computation time for the different models (in minutes).

Model	K									
	5	10	15	20	25	30	35	40	45	50
LSA	5.5	5.2	5.5	5.1	6.4	6.0	5.9	6.2	6.9	6.9
PLSA	241.6	257.3	291.7	369.9	492.5	758.0	1184.2	2097.4	4430.6	9537.7
LDA	7.3	7.6	7.8	8.0	8.5	8.8	9.2	9.7	10.0	10.3
LTR	8.2	8.4	8.7	9.0	9.1	9.3	10.0	10.2	10.7	11.0

is not suitable for use in large data environments, such as web documents and social network analysis. Since the underlying model of LTR is LDA, it can significantly reduce computational time as LDA, compared to PLSA. This means that LTR and LDA are suitable for use in large data environments. Compared with LDA, the performance improvement of LTR is between 14.356% and 19.823%, as shown in Fig. 4(d). This means that when LDA increases the time factor, its performance can be further improved.

Next, we perform some statistical analysis to analyze the differences in performance between different models. In this analysis, we use SPSS 14.0 for Windows to analyze the results of the experiment. Here we show only the results of Analysis of Variance (ANOVA) and Least Significant Difference (LSD) because all statistical analysis results are lengthy and tedious. The complete statistical analysis results are presented at http://hlcs.sytes.net/ltr/spss.htm. We first used ANOVA to analyze the difference analysis of average NSS values, which showed $F = 25{,}671.936$ (Table 6) is greater than or equal to $F_{0.05}(3,39,996) = 2.605131$ (F distribution), where $p < 0.05$. This provides strong evidence against the null hypothesis H_0, showing there is a significant difference in performance between the different models. We then conducted a pairwise comparison of the post LSD test, which is at the 5% significance level. Table 7 shows the results of LSD in pairwise comparisons of multiple mean values at $p < 0.05$. As shown in the LSD results, LTR performance is significantly better than other models in the NSS performance evaluation. Other post-test results also show the same trend that LTR's performance is significantly better than other models.

5.3. Cost evaluation of different semantic models

In this experiment, we want to compare the computation time needed for different models on a computer with an Intel 3.6 GHz CPU, 1600 MHz 16GB Memory, 240GB SSD hard drive. Table 8 shows the computation times for different models in different numbers of topics *Ks*. The cells in the table represent the computation time (in minutes) for each model under 1000 queries. LSA has the fastest computation time because it does not use any iterative method to estimate the relevant parameters. However, its performance is the worst because it can only find the query-related synonymous topics. PLSA has the slowest computation time because the EM algorithm grows exponentially with the number of terms and documents

Table 9
Blog08 dataset statistics.

Statistics	Size
Total number of unique posts	1,303,520
Average posts collected every day	38,557
Total number of permalinks	28,488,766
Total disk size of all unique posts	808GB
Total disk size of all permalinks	1445GB

(Hofmann, 2004). LDA and LTR use the Gibbs sampling algorithm rather than the EM algorithm to estimate the relevant parameters, so their computation time can be significantly faster than the PLSA (Speh et al., 2013).

Next, we analyze the meaning of Table 8 in detail. In LSA, the number of topics is only associated with the dimensionality reduction technique, which preserves the K noise-free topics and sets the remaining topics to zero. Since the correlation between the time needed for dimensionality reduction and the size of K is low, the computation time of LSA is only slightly increased by 1.4 (6.9−5.5) min as K increases from 5 to 50. However, compared to other models, the correlation between the computational time of PLSA and the size of K is greatest. Based on the results in the table, we found that PLSA's computation time increased from 241.6 min to 9537.7 min when K increased from 5 to 50. This represents the time needed for PLSA to grow exponentially with increasing K. This is consistent with Chen's discovery (Chen, 2012), that is, the time needed for the EM algorithm increases exponentially as the number of topics increases. When K increases from 5 to 50, the computation time of LDA and LTR increases by 3 min (10.3−7.3) and 2.8 (11.0−8.2) min, respectively. That is, for 1000 queries, the average increase in the computation time for each query will not exceed 0.18 s. The reason is that the correlation between the time needed for the Gibbs sampling algorithm and the size of K is not obvious. According to the results in Fig. 4(d) and Table 8, we found that when LDA considers the time factor, the extra time for each query would not exceed 0.06 (1 min/1000) s. However, this factor can improve the performance of LDA by at least 14.456%. This means that for the LDA model, the time factor only needs to add a small amount of computation time, but it can significantly improve performance.

5.4. Performance comparison of LDA-based time topic models

In this experiment, we analyze and compare the performance of different LDA-based time topic models. These comparative models include LDA, TOT, S-ATM, Twitter-LDA, TM-LDA, and LTR for this study. The practice and focus of all models (except LTR) are described in Section 2.3. Since this study analyzes the performance of blog searches, we use the Blog08 collection (Macdonald, Ounis, & Soboroff, 2009) in TREC BLOG Track as a data set for different models. The data set is a rich blog database that has collected 1,303,520 unique blog posts from January 14, 2008 to February 10, 2009. Table 9 is the relevant statistical report for this data set.

All models are designed to predict the topic distribution of future posts based on historical posts. Therefore, we use the perplexity metric to evaluate all models against the actual word occurrences in future posts. Often, the metric is used to measure how well a topic model fits the word distribution of a data set. It is defined as follows:

$$Perplexity_m = 2^{-\sum_{i=1}^{w} log_2 P_m(x_i)} \tag{15}$$

where $m \in$ {LDA, TOT, S-ATM, Twitter-LDA, TM-LDA, LTR} is a comparison model, $Perplexity_m$ represents the perplexity of model m, w is the number of words in the document, $P_m(x_i)$ is the probability of the occurrence of word x_i estimated by the model m. Intuitively, if the model yields higher probability for the occurrences of words in the document than words that are not in the document, the model is more accurate and the perplexity will be lower.

Fig. 5 is the perplexity distribution for different models. According to the results in the figure, we found that the perplexity of LDA is between 1401 and 1481. Unlike other models, LDA cannot effectively identify changes in hot topics in different time zones. The main reason is that LDA does not take any time parameters. Although it can identify different topics, it cannot effectively identify the trend of changing topics with time shift.

Next, we compare other LDA-based time topic models. Based on the results shown in the figure, we found that the lowest perplexity value falls between 2008/12/14 and 2009/1/14. The reason is that both Christmas and New Year have the most posts in this period to discuss how to arrange holiday activities. The second and third lowest perplexity values fall between 2008/7/14 and 2008/8/14 and between 2008/8/14 and 2008/9/14. The reason is that there are many posts during these two periods that discuss the news and achievements of the 2008 Summer Olympics (time from 2008/8/8 to 2008/8/24). The fourth and fifth lowest perplexity values fall between 2008/10/14 and 2008/11/14 and between 2008/9/14 and 2008/10/14. The reason is that there are many posts during these two periods that discuss the relevant news and election results of the US presidential election (2008/11/4). The trend of these results is that when there are more posts over a period discussing some of the relevant topics, the discussion of the relevant topics will tend to be consistent.

The average perplexity value in Twitter-LDA is the highest in all models. The reason includes two points. First, the model uses a semi-automatic way to group all topics. This way can only classify existing topics, that is, it cannot handle new topics that often appear on the social network. Second, all the topics used in the model are based on the topics defined in the

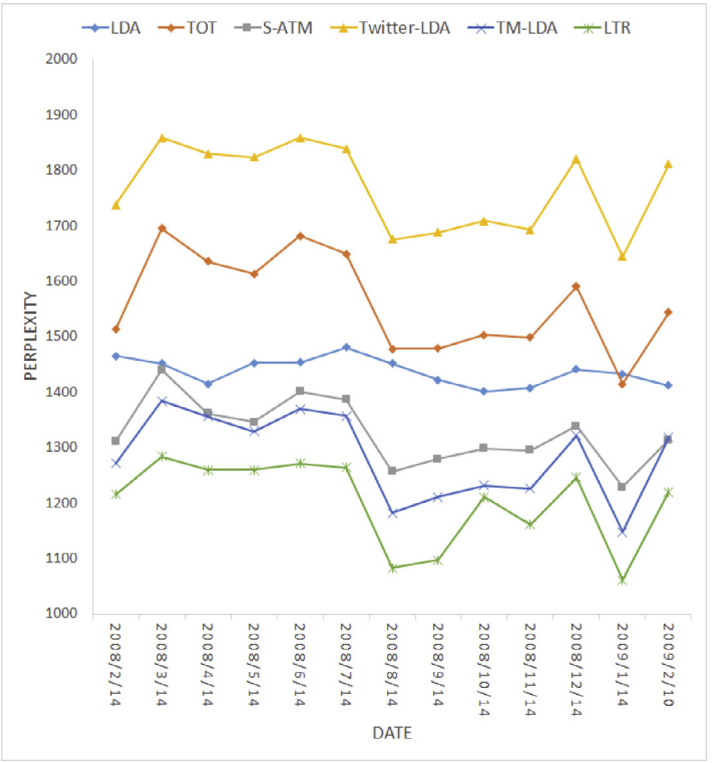


Fig. 5. The perplexity distribution of different models.

existing corpus. However, this will result in the need to manually classify all the topics that do not exist in the corpus. That is, it cannot handle the real-time topics on the social network. The second highest model for the average perplexity value is TOT. The reason includes two points. First, the time parameter of the model is based on the time at which a particular topic occurs. However, it is necessary to predefine the relevant topics to perform follow-up document clustering. Second, the model's time parameter is selected based on two states. This means that the model cannot effectively handle the time with multiple states and continuous characteristics. The third highest model for the average perplexity value is S-ATM. The reason includes two points. First, the model uses some predefined topics to analyze trends in all topics. However, it cannot analyze topics that do not exist in predefined topics. Second, the model uses a discrete event to describe the time segment of the topics. This makes it difficult to reflect the continuous trend of topic changes. The fourth highest model for the average perplexity value is TM-LDA. The reason includes two points. First, the model uses a long time to train the relevant training parameters. This means that it cannot dynamically predict related topics based on different document distributions. Second, the parameters are the overall trend for all documents. That is, it cannot dynamically predict the distribution of different topics based on different document sets. LTR has the lowest average perplexity value. The reason consists of three points. First, LTR does not need to predefine related topics. Because each query contains posts that are not the same, we cannot infer the trend of all topics by predefined topics. Second, it uses a real number to define the time parameter. This means that it can reflect the continuing trend of topic change in different periods. Third, it uses the Dirichlet distribution to estimate the relationship between topic and time. Since the Dirichlet distribution is a conjugate probability distribution of the multinomial distribution, it means that it can describe the trend of all topics in different states.

5.5. Discussions on data sets and performance indicators used by different models

In Section 5, we used different test data sets and different performance indicators for different experimental types. In this subsection, we discuss in detail the reasons for using different data sets and indicators for different experimental types. Table 10 is a comparison of the use of data sets and indicators for different models.

In this study, we divided the experimental types into general semantic and LDA-based semantic based on data processing units. The main data processing units of general and LDA-based semantic types are keywords and topics, respectively. In the general semantic type, the model we evaluated includes LSA, PLSA, LDA, and LTR. There are three reasons why Google Blog Search is used as a test data set for this type. The first reason is that the data set is collected from the perspective of keywords. The second reason is that it only calculates the similarity between keywords in the first 350 posts for each query. Although there are many keywords included in the 350 posts, the keywords that need to be processed can be significantly

Table 10
Comparison of data sets and indicators for different models.

Experimental type (Models)	Test data set Indicator	Reason Reason
General semantic (LSA, PLSA, LDA, LTR)	Google Blog Search	• The collection of this data set is based on keywords • Less data for each keyword • Shorter analysis time for each keyword
	NSS	• Based on keyword similarity • The distance between keywords • No predefined keywords are needed
LDA-based semantic (LDA, TOT, S-ATM, Twitter-LDA, TM-LDA, LTR)	TREC Blog08 Track	• The collection of this data set is based on topics • More data for each topic • Longer analysis time for each topic
	Perplexity	• Based on topic dispersion • The convergence within the topic • Predefined topics are needed

reduced by the preprocessing and NLP steps in this study. That is, there is less data to process for each keyword. The third reason is that when the needed processing data is less, the time needed for analyzing the keyword similarity is often shorter. Therefore, in our experimental system, we can immediately calculate the similarity between keywords. There are three reasons why NSS is used as a performance indicator for this type. The first reason is that the indicator calculates the semantic similarity between two keywords. The second reason is that it evaluates the distance between two keywords. Based on the first and second reasons, we found that the smaller the distance between the two keywords, the closer the semantics between them. The third reason is that it does not need any predefined training data. That is, it does not need to predefine any keywords that need to be processed.

In the LDA-based semantic type, the model we evaluated includes LDA, TOT, S-ATM, Twitter-LDA, TM-LDA, and LTR. There are three reasons why TREC Blog08 Track is used as a test data set for this type. The first reason is that this data set is collected from the perspective of topics. The posts collected by this data set is based on certain topics as seeds (Gerani, Keikha, Carman, & Crestani, 2011; Zhang, Qiu, Bu, Qu, & Chen, 2009). Because the number of topics used in this data set is limited, the number of posts for each topic is huge. That is, the second reason is that the data handled by each topic is more. Since each topic has more data, the third reason is that the processing time for each topic will be longer. Therefore, in our experimental results, we can only use the batch method to calculate the perplexity between topics. There are three reasons why perplexity is used as a performance indicator for this type. The first reason is that the indicator is used to calculate the dispersion between topics. The second reason is that it evaluates the convergence within the topic. Based on the first and second reasons, we found that the posts within the topic were more concentrated, the lower the degree of dispersion between topics. The third reason is that it needs some predefined training topics to facilitate calculating the perplexity of posts within the topic.

6. Conclusions and future work

Since the discussion topic of blog posts is closely related to the update time of posts, how to effectively add time to blog search becomes an important task. In this paper, we add the time parameter to the blog topic to distinguish the differences in the same topic at different times. The advantage of adding this time parameter to posts is that we can add a small amount of computation time to significantly improve blog search performance. This study has two main contributions. The first is that we build a system that considers the importance of blog topics at different times. The second is that we add the time factor to the post, which can rank the blog topic based on the popularity of the post.

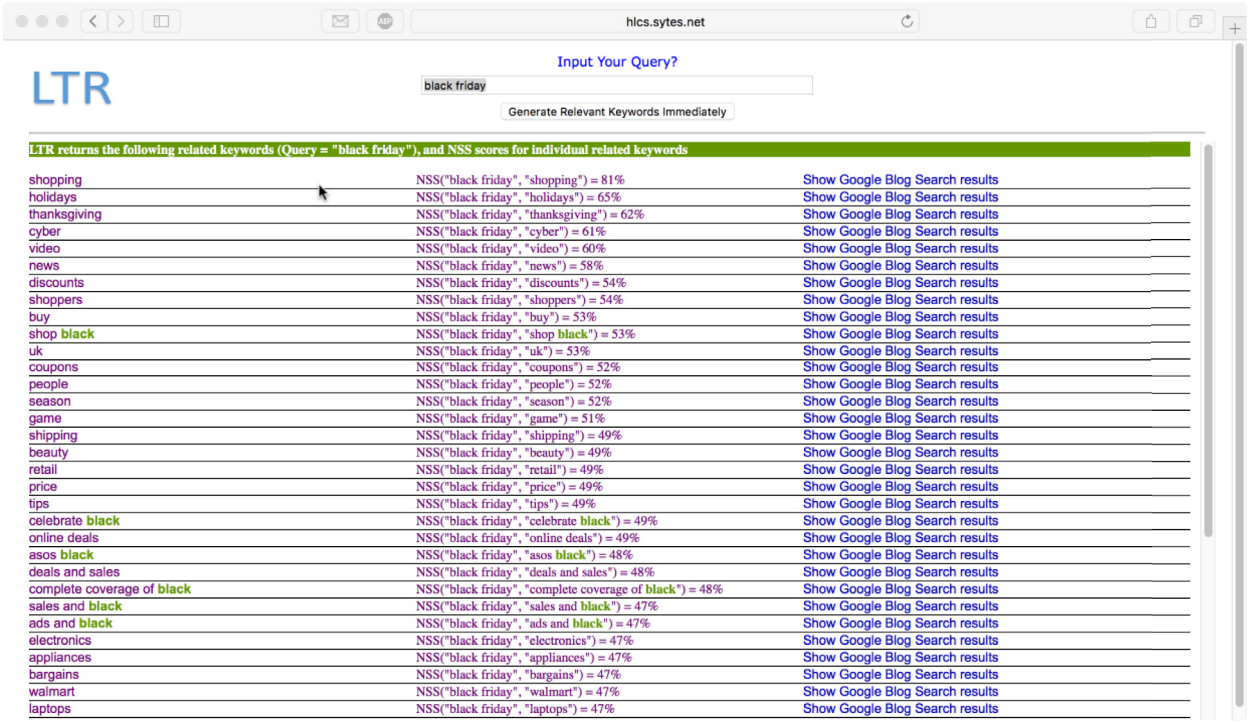
In the future, we plan to develop a general semantic model that can consider a wide variety of document relationships. Through the model, we expect to deal with various latent topic relationships in the document. When different relationships are added to the model, it needs more computational time to estimate the distribution of different relationships and MLE. Therefore, we also plan to combine the different relationships with similar features into a relationship to reduce the computational time needed.

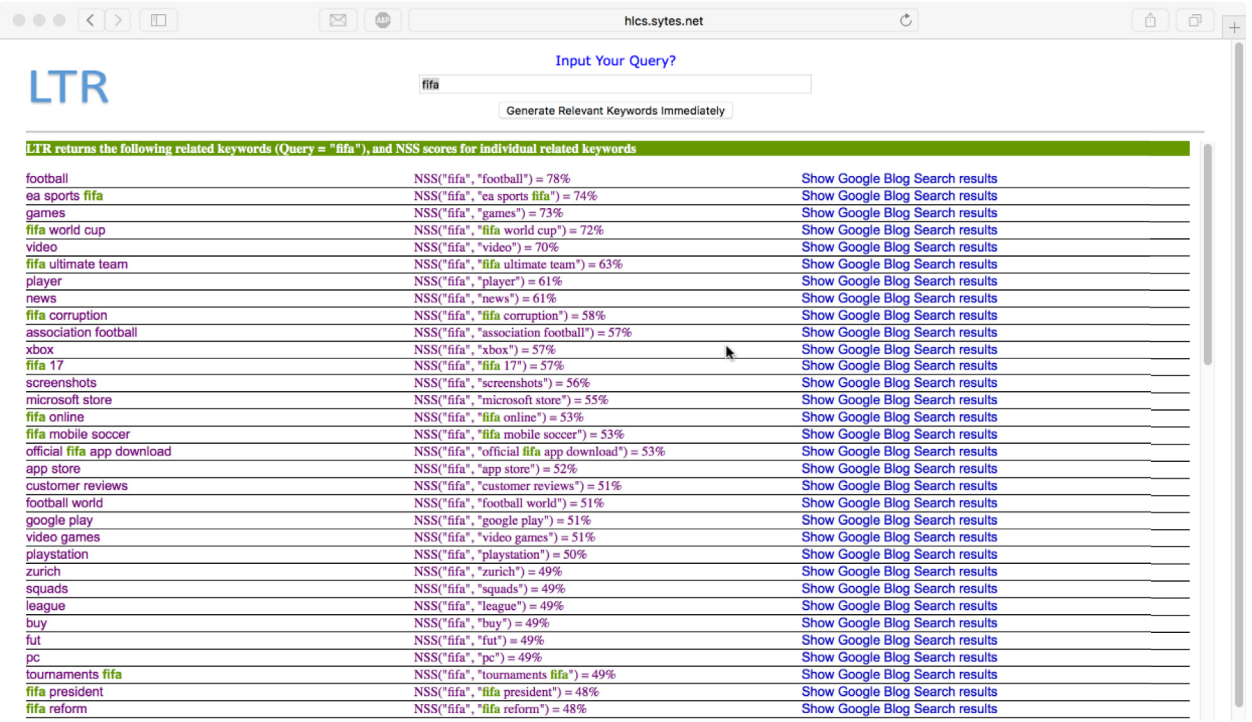
Acknowledgments

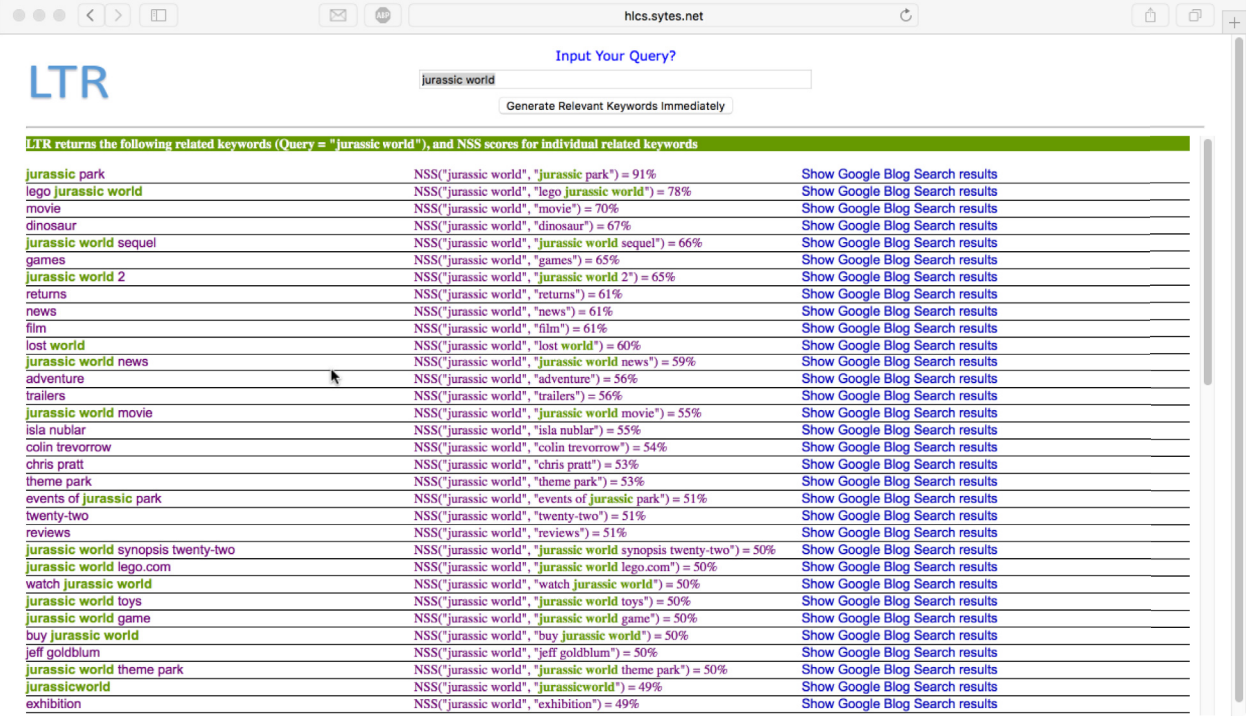
We would like to thank anonymous reviewers of the paper for their constructive comments, which help us to improve the paper in several ways. This work was supported in part by Ministry of Science and Technology, Taiwan under Grant Most 106-2410-H-259-011 & 105-2221-E-259-030.

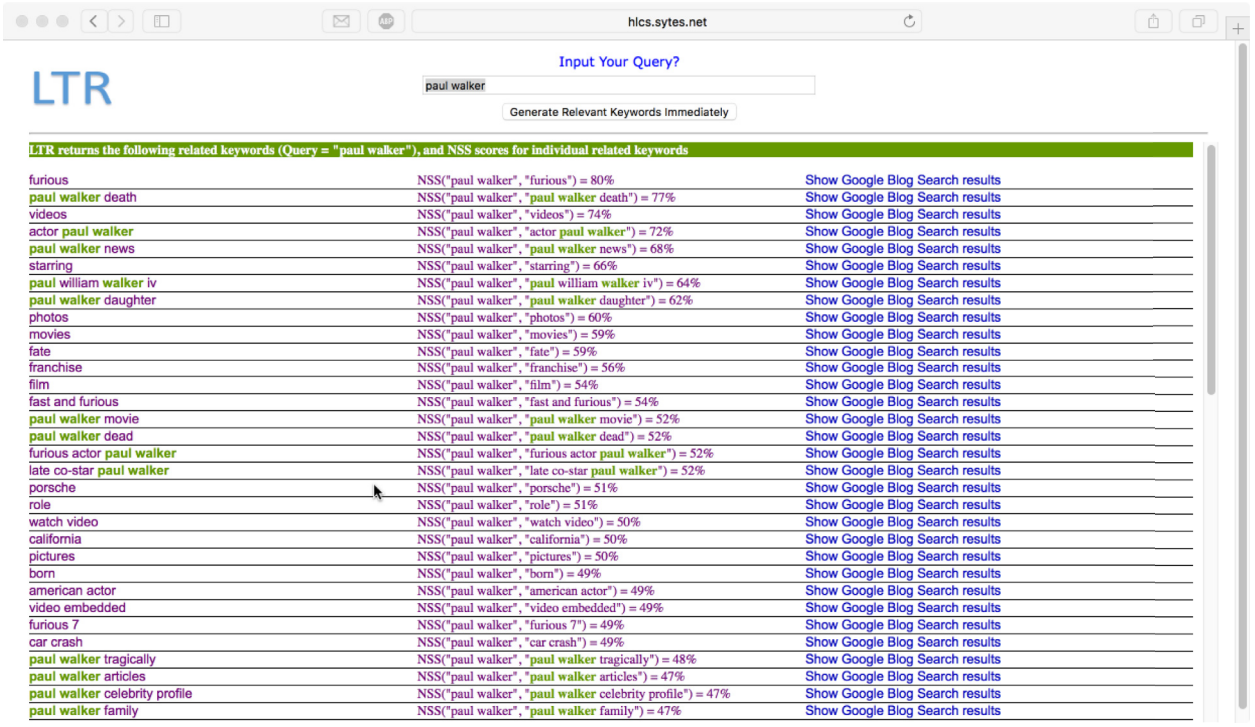
Appendix

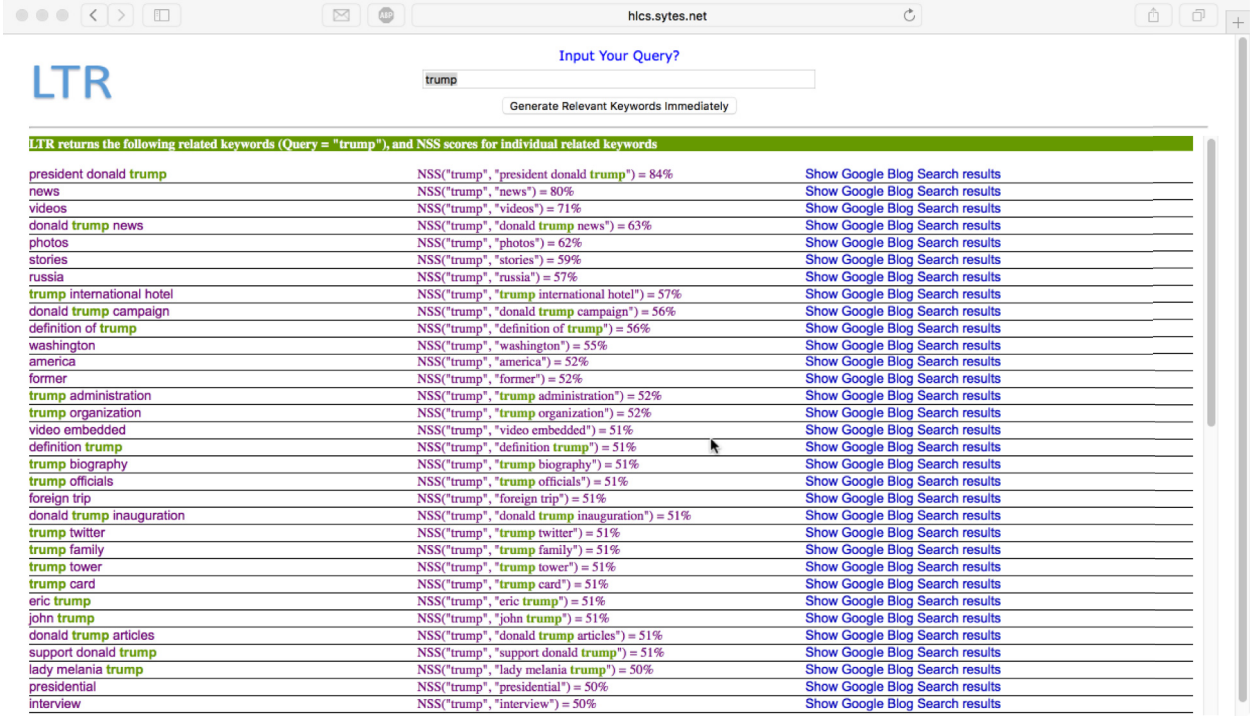
We randomly selected five queries (“black friday”, “fifa”, “jurassic world”, “paul walker”, “trump”) to run our system from 1000 test queries (access date is 2017/5/20) and the results are shown below.




hlcs.sytes.net
LTR
Input Your Query?
fifa
Generate Relevant Keywords Immediately
LTR returns the following related keywords (Query = "fifa"), and NSS scores for individual related keywords
football NSS("fifa", "football") = 78% Show Google Blog Search results
ea sports fifa NSS("fifa", "ea sports fifa") = 74% Show Google Blog Search results
games NSS("fifa", "games") = 73% Show Google Blog Search results
fifa world cup NSS("fifa", "fifa world cup") = 72% Show Google Blog Search results
video NSS("fifa", "video") = 70% Show Google Blog Search results
fifa ultimate team NSS("fifa", "fifa ultimate team") = 63% Show Google Blog Search results
player NSS("fifa", "player") = 61% Show Google Blog Search results
news NSS("fifa", "news") = 61% Show Google Blog Search results
fifa corruption NSS("fifa", "fifa corruption") = 58% Show Google Blog Search results
association football NSS("fifa", "association football") = 57% Show Google Blog Search results
xbox NSS("fifa", "xbox") = 57% Show Google Blog Search results
fifa 17 NSS("fifa", "fifa 17") = 57% Show Google Blog Search results
screenshots NSS("fifa", "screenshots") = 56% Show Google Blog Search results
microsoft store NSS("fifa", "microsoft store") = 55% Show Google Blog Search results
fifa online NSS("fifa", "fifa online") = 53% Show Google Blog Search results
fifa mobile soccer NSS("fifa", "fifa mobile soccer") = 53% Show Google Blog Search results
official fifa app download NSS("fifa", "official fifa app download") = 53% Show Google Blog Search results
app store NSS("fifa", "app store") = 52% Show Google Blog Search results
customer reviews NSS("fifa", "customer reviews") = 51% Show Google Blog Search results
football world NSS("fifa", "football world") = 51% Show Google Blog Search results
google play NSS("fifa", "google play") = 51% Show Google Blog Search results
video games NSS("fifa", "video games") = 51% Show Google Blog Search results
playstation NSS("fifa", "playstation") = 50% Show Google Blog Search results
zurich NSS("fifa", "zurich") = 49% Show Google Blog Search results
squads NSS("fifa", "squads") = 49% Show Google Blog Search results
league NSS("fifa", "league") = 49% Show Google Blog Search results
buy NSS("fifa", "buy") = 49% Show Google Blog Search results
fut NSS("fifa", "fut") = 49% Show Google Blog Search results
pc NSS("fifa", "pc") = 49% Show Google Blog Search results
tournaments fifa NSS("fifa", "tournaments fifa") = 49% Show Google Blog Search results
fifa president NSS("fifa", "fifa president") = 48% Show Google Blog Search results
fifa reform NSS("fifa", "fifa reform") = 48% Show Google Blog Search results


hlcs.sytes.net
LTR
Input Your Query?
jurassic world
Generate Relevant Keywords Immediately
LTR returns the following related keywords (Query = "jurassic world"), and NSS scores for individual related keywords
jurassic park NSS("jurassic world", "jurassic park") = 91% Show Google Blog Search results
lego jurassic world NSS("jurassic world", "lego jurassic world") = 78% Show Google Blog Search results
movie NSS("jurassic world", "movie") = 70% Show Google Blog Search results
dinosaur NSS("jurassic world", "dinosaur") = 67% Show Google Blog Search results
jurassic world sequel NSS("jurassic world", "jurassic world sequel") = 66% Show Google Blog Search results
games NSS("jurassic world", "games") = 65% Show Google Blog Search results
jurassic world 2 NSS("jurassic world", "jurassic world 2") = 65% Show Google Blog Search results
returns NSS("jurassic world", "returns") = 61% Show Google Blog Search results
news NSS("jurassic world", "news") = 61% Show Google Blog Search results
film NSS("jurassic world", "film") = 61% Show Google Blog Search results
lost world NSS("jurassic world", "lost world") = 60% Show Google Blog Search results
jurassic world news NSS("jurassic world", "jurassic world news") = 59% Show Google Blog Search results
adventure NSS("jurassic world", "adventure") = 56% Show Google Blog Search results
trailers NSS("jurassic world", "trailers") = 56% Show Google Blog Search results
jurassic world movie NSS("jurassic world", "jurassic world movie") = 55% Show Google Blog Search results
isla nublar NSS("jurassic world", "isla nublar") = 55% Show Google Blog Search results
colin trevorrow NSS("jurassic world", "colin trevorrow") = 54% Show Google Blog Search results
chris pratt NSS("jurassic world", "chris pratt") = 53% Show Google Blog Search results
theme park NSS("jurassic world", "theme park") = 53% Show Google Blog Search results
events of jurassic park NSS("jurassic world", "events of jurassic park") = 51% Show Google Blog Search results
twenty-two NSS("jurassic world", "twenty-two") = 51% Show Google Blog Search results
reviews NSS("jurassic world", "reviews") = 51% Show Google Blog Search results
jurassic world synopsis twenty-two NSS("jurassic world", "jurassic world synopsis twenty-two") = 50% Show Google Blog Search results
jurassic world lego.com NSS("jurassic world", "jurassic world lego.com") = 50% Show Google Blog Search results
watch jurassic world NSS("jurassic world", "watch jurassic world") = 50% Show Google Blog Search results
jurassic world toys NSS("jurassic world", "jurassic world toys") = 50% Show Google Blog Search results
jurassic world game NSS("jurassic world", "jurassic world game") = 50% Show Google Blog Search results
buy jurassic world NSS("jurassic world", "buy jurassic world") = 50% Show Google Blog Search results
jeff goldblum NSS("jurassic world", "jeff goldblum") = 50% Show Google Blog Search results
jurassic world theme park NSS("jurassic world", "jurassic world theme park") = 50% Show Google Blog Search results
jurassicworld NSS("jurassic world", "jurassicworld") = 49% Show Google Blog Search results
exhibition NSS("jurassic world", "exhibition") = 49% Show Google Blog Search results


hlcs.sytes.net
LTR
Input Your Query?
paul walker
Generate Relevant Keywords Immediately
LTR returns the following related keywords (Query = "paul walker"), and NSS scores for individual related keywords
furious NSS("paul walker", "furious") = 80% Show Google Blog Search results
paul walker death NSS("paul walker", "paul walker death") = 77% Show Google Blog Search results
videos NSS("paul walker", "videos") = 74% Show Google Blog Search results
actor paul walker NSS("paul walker", "actor paul walker") = 72% Show Google Blog Search results
paul walker news NSS("paul walker", "paul walker news") = 68% Show Google Blog Search results
starring NSS("paul walker", "starring") = 66% Show Google Blog Search results
paul william walker iv NSS("paul walker", "paul william walker iv") = 64% Show Google Blog Search results
paul walker daughter NSS("paul walker", "paul walker daughter") = 62% Show Google Blog Search results
photos NSS("paul walker", "photos") = 60% Show Google Blog Search results
movies NSS("paul walker", "movies") = 59% Show Google Blog Search results
fate NSS("paul walker", "fate") = 59% Show Google Blog Search results
franchise NSS("paul walker", "franchise") = 56% Show Google Blog Search results
film NSS("paul walker", "film") = 54% Show Google Blog Search results
fast and furious NSS("paul walker", "fast and furious") = 54% Show Google Blog Search results
paul walker movie NSS("paul walker", "paul walker movie") = 52% Show Google Blog Search results
paul walker dead NSS("paul walker", "paul walker dead") = 52% Show Google Blog Search results
furious actor paul walker NSS("paul walker", "furious actor paul walker") = 52% Show Google Blog Search results
late co-star paul walker NSS("paul walker", "late co-star paul walker") = 52% Show Google Blog Search results
porsche NSS("paul walker", "porsche") = 51% Show Google Blog Search results
role NSS("paul walker", "role") = 51% Show Google Blog Search results
watch video NSS("paul walker", "watch video") = 50% Show Google Blog Search results
california NSS("paul walker", "california") = 50% Show Google Blog Search results
pictures NSS("paul walker", "pictures") = 50% Show Google Blog Search results
born NSS("paul walker", "born") = 49% Show Google Blog Search results
american actor NSS("paul walker", "american actor") = 49% Show Google Blog Search results
video embedded NSS("paul walker", "video embedded") = 49% Show Google Blog Search results
furious 7 NSS("paul walker", "furious 7") = 49% Show Google Blog Search results
car crash NSS("paul walker", "car crash") = 49% Show Google Blog Search results
paul walker tragically NSS("paul walker", "paul walker tragically") = 48% Show Google Blog Search results
paul walker articles NSS("paul walker", "paul walker articles") = 47% Show Google Blog Search results
paul walker celebrity profile NSS("paul walker", "paul walker celebrity profile") = 47% Show Google Blog Search results
paul walker family NSS("paul walker", "paul walker family") = 47% Show Google Blog Search results


hlcs.sytes.net
LTR
Input Your Query?
trump
Generate Relevant Keywords Immediately
LTR returns the following related keywords (Query = "trump"), and NSS scores for individual related keywords
president donald trump NSS("trump", "president donald trump") = 84% Show Google Blog Search results
news NSS("trump", "news") = 80% Show Google Blog Search results
videos NSS("trump", "videos") = 71% Show Google Blog Search results
donald trump news NSS("trump", "donald trump news") = 63% Show Google Blog Search results
photos NSS("trump", "photos") = 62% Show Google Blog Search results
stories NSS("trump", "stories") = 59% Show Google Blog Search results
russia NSS("trump", "russia") = 57% Show Google Blog Search results
trump international hotel NSS("trump", "trump international hotel") = 57% Show Google Blog Search results
donald trump campaign NSS("trump", "donald trump campaign") = 56% Show Google Blog Search results
definition of trump NSS("trump", "definition of trump") = 56% Show Google Blog Search results
washington NSS("trump", "washington") = 55% Show Google Blog Search results
america NSS("trump", "america") = 52% Show Google Blog Search results
former NSS("trump", "former") = 52% Show Google Blog Search results
trump administration NSS("trump", "trump administration") = 52% Show Google Blog Search results
trump organization NSS("trump", "trump organization") = 52% Show Google Blog Search results
video embedded NSS("trump", "video embedded") = 51% Show Google Blog Search results
definition trump NSS("trump", "definition trump") = 51% Show Google Blog Search results
trump biography NSS("trump", "trump biography") = 51% Show Google Blog Search results
trump officials NSS("trump", "trump officials") = 51% Show Google Blog Search results
foreign trip NSS("trump", "foreign trip") = 51% Show Google Blog Search results
donald trump inauguration NSS("trump", "donald trump inauguration") = 51% Show Google Blog Search results
trump twitter NSS("trump", "trump twitter") = 51% Show Google Blog Search results
trump family NSS("trump", "trump family") = 51% Show Google Blog Search results
trump tower NSS("trump", "trump tower") = 51% Show Google Blog Search results
trump card NSS("trump", "trump card") = 51% Show Google Blog Search results
eric trump NSS("trump", "eric trump") = 51% Show Google Blog Search results
john trump NSS("trump", "john trump") = 51% Show Google Blog Search results
donald trump articles NSS("trump", "donald trump articles") = 51% Show Google Blog Search results
support donald trump NSS("trump", "support donald trump") = 51% Show Google Blog Search results
lady melania trump NSS("trump", "lady melania trump") = 50% Show Google Blog Search results
presidential NSS("trump", "presidential") = 50% Show Google Blog Search results
interview NSS("trump", "interview") = 50% Show Google Blog Search results

References

Batra, S., & Bawa, S. (2010). Web service categorization using normalized similarity score. *International Journal of Computer Theory and Engineering, 2*(1), 139–142.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research, 3*(1), 993–1022.

Bolelli, L., Ertekin, Ş., & Giles, C. L. (2009). Topic and trend detection in text collections using latent Dirichlet allocation. In *Proceedings of the 31th European conference on IR research on advances in information retrieval, Toulouse, Springer Press, France* (pp. 776–780).

Brahmane, A. V., & Amune, A. (2014). A survey of dynamic distributed network intrusion detection using online adaboost-based parameterized methods. *International Journal of Innovative Research in Advanced Engineering, 1*(9), 256–262.

Chandramohan, D., Khapre, S., & Ashokkumar, S. (2011). A study of finding similarities in web service using metrics. *International Journal of Scientific and Engineering Research, 2*(6), 1–7.

Chen, L.-C. (2012). Building a term suggestion and ranking system based on a probabilistic analysis model and a semantic analysis graph. *Decision Support Systems, 53*(1), 257–266.

Cilibrasi, R. L., & Vitányi, P. M. B. (2007). The Google similarity distance. *IEEE Transaction On Knowledge and Data Engineering, 19*(3), 370–383.

Cosma, G., & Joy, M. (2012). An approach to source-code plagiarism detection and investigation using latent semantic analysis. *IEEE Transactions on Computers, 61*(3), 379–394.

Dumais, S. T. (2004). Latent semantic analysis. *Annual Review of Information Science and Technology, 38*(1), 189–230.

Fernandez-Beltran, R., & Pla, F. (2015). Incremental probabilistic latent semantic analysis for video retrieval. *Image and Vision Computing, 38*(C), 1–12.

Fox, C. (1989). A stop list for general text. *ACM SIGIR Forum, 24*(1-2), 19–35.

Fujimura, K., Toda, H., Inoue, T., Hiroshima, N., Kataoka, R., & Sugizaki, M. (2006). BLOGRANGER-a multi-faceted blog search engine. In *Proceedings of the WWW 2006 workshop on the weblogging ecosystem: aggregation, analysis and dynamics, Edinburgh, UK W3C* (pp. 22–26).

Gerani, S., Keikha, M., Carman, M., & Crestani, F. (2011). Personal Blog Retrieval using Opinion Features. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval* (pp. 747–750). Dublin, Ireland: Springer Press.

Geyer, W., & Dugan, C. (2010). Inspired by the audience –a topic suggestion system for blog writers and readers. In *Proceedings of the 2010 ACM conference on computer supported cooperative work, Savannah, GA, USA, ACM* (pp. 237–240).

Google. (2014). *Google stop-words* Retrieved 29 May 2017. https://code.google.com/archive/p/stop-words/.

Hazel, P. (2017). *PCRE - perl compatible regular expressions* Retrieved 29 May 2017. http://www.pcre.org/.

Hirst, G., & St-Onge, D. (1998). Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database, 305*(1), 305–322.

Hofmann, T. (2001). Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning, 42*(1), 177–196.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems, 22*(1), 89–115.

Hofmann, T., Schölkopf, B., & Smola, A. J. (2008). Kernel methods in machine learning. *The Annals of Statistics, 36*(3), 1171–1220.

Hsieh, J.-W., Chen, L.-C., Chen, S.-Y., Chen, D.-Y., Alghyaline, S., & Chiang, H.-F. (2015). Vehicle color classification under different lighting conditions through color correction. *IEEE Sensors Journal, 15*(2), 971–983.

Husby, S. D., & Barbosa, D. (2012). Topic classification of blog posts using distant supervision. In *Proceedings of the 13th conference of the european chapter of the association for computational linguistics, Avignon, ACM, France* (pp. 28–36).

Jeong, O.-R., & Oh, J. (2012). Social community based blog search framework. In *Proceedings of the 17th international conference on database systems for advanced applications, Busan, South Korea Springer Press* (pp. 130–141).

Ji, Z., Jing, P., Wang, J., & Su, Y. (2012). Scene image classification with biased spatial block and PLSA. *International Journal of Digital Content Technology and its Applications, 6*(1), 398–404.

Jiang, J. J., & Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of International Conference on Research in Computational Linguistics (ROCLING X)* (pp. 19–33). Taiwan: Sinica Taiwan.

Keikha, M., Crestani, F., & Carman, M. J. (2013). Searching blog sites with product reviews. In *Proceedings of the 15th international conference on human interface and the management of information: information and interaction for learning, culture, collaboration and business - Volume Part III, Las Vegas, NV Springer Press* (pp. 495–500).

Keikha, M., Gerani, S., & Crestani, F. (2011). Time-based relevance models. In *Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval, Beijing, China, ACM* (pp. 1087–1088).

Kim, J., Yun, U., Pyun, G., Ryang, H., Lee, G., Yoon, E., et al. (2015). A blog ranking algorithm using analysis of both blog influence and characteristics of blog posts. *Cluster Computing, 18*(1), 157–164.

Klein, R., Kyrilov, A., & Tokman, M. (2011). Automated assessment of short free-text responses in computer science using latent semantic analysis. In *Proceedings of the 16th annual joint conference on innovation and technology in computer science education, Darmstadt, ACM, German* (pp. 158–162).

Krestel, R., Fankhauser, P., & Nejdl, W. (2009). Latent Dirichlet allocation for tag recommendation. In *Proceedings of the 3rd ACM conference on recommender systems* (pp. 61–68). New York: ACM.

Kuo, F.-F., Shan, M.-K., & Lee, S.-Y. (2013). Background music recommendation for video based on multimodal latent semantic analysis. In *Proceedings of the 2013 IEEE international conference on multimedia and expo, San Jose, USA IEEE* (pp. 1–6).

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes, 25*(1), 259–284.

Landauer, T. K., McNamara, D. S., Dennis, S., & Kintsch, W. (2013). *Handbook of latent semantic analysis*. London, UK: Psychology Press.

Leacock, C., & Chodorow, M. (1998). Combining local context and WordNet similarity for word sense identification. *WordNet: An electronic lexical database, 49*(2), 265–283.

Li, M., Li, W. K., & Li, G. (2013). On mixture memory garch models. *Journal of Time Series Analysis, 34*(6), 606–624.

Liénou, M., Maître, H., & Datcu, M. (2010). Semantic annotation of satellite images using latent Dirichlet allocation. *IEEE Geoscience and Remote Sensing Letters, 7*(1), 28–32.

Lin, D. (1998). An Information-Theoretic Definition of Similarity. In *Proceedings of the 15th International Conference on Machine Learning* (pp. 296–304). Wisconsin, USA: Morgan Kaufmann.

Lindsey, R., Veksler, V. D., Grintsvayg, A., & Gray, W. D. (2007). Be wary of what your computer reads: the effects of corpus selection on measuring semantic relatedness. In *Proceedings of the 8th international conference on cognitive modeling, Ann Arbor, Michigan Taylor & Francis Press* (pp. 279–284).

Lintean, M., Moldovan, C., Rus, V., & McNamara, D. (2010). The role of local and global weighting in assessing the semantic similarity of texts using latent semantic analysis. In *Proceedings of the 23th international Florida artificial intelligence research society conference, Marco Island, Florida AAAI Press* (pp. 235–240).

Liu, Y.-H., Chen, Y.-L., & Ho, W.-L. (2015). Predicting associated statutes for legal problems. *Information Processing and Management, 51*(1), 194–211.

Liu, Z., Zhang, Y., Chang, E. Y., & Sun, M. (2011). PLDA+: parallel latent Dirichlet allocation with data placement and pipeline processing. *ACM Transactions on Intelligent Systems and Technology, 2*(3) 26:21–26:18.

Logan, B., Kositsky, A., & Moreno, P. (2004). Semantic analysis of song lyrics. In *Proceedings of the 2004 IEEE international conference on multimedia and expo, Taipei, IEEE, Taiwan* (pp. 827–830).

Luh, C.-J., Yang, S.-A., & Huang, T.-L. D. (2016). Estimating Google's search engine ranking function from a search engine optimization perspective. *Online Information Review, 40*(2), 239–255.

Lukins, S. K., Kraft, N. A., & Etzkorn, L. H. (2008). Source code retrieval for bug localization using latent Dirichlet allocation. In *Proceedings of the 2008 15th working conference on reverse engineering, Antwerp, IEEE, Belgium* (pp. 155–164).

Macdonald, C., Ounis, I., & Soboroff, I. (2009). Overview of the TREC-2009 blog track. In *Proceedings of TREC 2009, Gaithersburg, MD National Institute of Standards and Technology* (pp. 1–9).
McInerney, J., Rogers, A., & Jennings, N. R. (2012). Improving location prediction services for new users with probabilistic latent semantic analysis. In *Proceedings of the 2012 ACM conference on ubiquitous computing, Pittsburgh, Pennsylvania ACM* (pp. 906–910).
Mesaros, A., Heittola, T., & Klapuri, A. (2011). Latent semantic analysis in sound event detection. In *Proceedings of the 19th European signal processing conference, Barcelona, EURASIP, Spain* (pp. 1307–1311).
Mishne, G., & Rijke, M. d. (2006). A study of blog search. In *Proceedings of the 28th European conference on advances in information retrieval, London, Springer Press, UK* (pp. 289–301).
MySQL. (2017). *MySQL full-text stopwords* Retrieved 29 May 2017. https://dev.mysql.com/doc/refman/5.5/en/fulltext-stopwords.html.
Ozsoy, M. G., Alpaslan, F. N., & Cicekli, I. (2011). Text summarization using latent semantic analysis. *Journal of Information Science, 37*(4), 405–417.
Patwardhan, S., Banerjee, S., & Pedersen, T. (2003). Using measures of semantic relatedness for word sense disambiguation. In *Proceedings of the international conference on intelligent text processing and computational linguistics, Mexico City, Springer Press, Mexico* (pp. 241–257).
Pingdom. (2015). *2015 the web shown in numbers* Retrieved 29 May 2017. https://goo.gl/mW77a3.
Porter, M., & Boulton, R. (2017). *Snowball: a language for stemming algorithms* Retrieved 29 May 2017. http://snowball.tartarus.org/.
Prayiush. (2015). *Number of blogs up from 35 million in 2006 to 181 million by the end of 2011* Retrieved 29 May 2017. https://goo.gl/8WLITs.
Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th international joint conference on Artificial intelligence* (pp. 448–453). Montreal, Quebec, Canada: Morgan Kaufmann.
Shi, C., Quan, J., & Li, M. (2013). Information extraction for computer science academic rankings system. In *Proceedings of the 2013 international conference on cloud and service computing, Beijing, China, IEEE* (pp. 69–76).
Siddiqui, A., Mishra, N., & Verma, J. S. (2015). A survey on automatic image annotation and retrieval. *International Journal of Computer Applications, 118*(20), 27–32.
Somasundaram, K., & Murphy, G. C. (2012). Automatic categorization of bug reports using latent Dirichlet allocation. In *Proceedings of the 5th India software engineering conference, Kanpur, ACM, India* (pp. 125–130).
Speh, J., Muhic, A., & Rupnik, J. (2013). Parameter estimation for the latent dirichlet allocation. In *Proceedings of the 2013 conference on data mining and data warehouses, Ljubljana, Information Society, Slovenia* (pp. 1–4).
Takama, Y., Kajinami, T., & Matsumura, A. (2005). Blog search with keyword map-based relevance feedback. In *Proceedings of the 2nd international conference on fuzzy systems and knowledge discovery - Volume Part II, Changsha, Springer Press, China* (pp. 1208–1215).
Thelwall, M., & Hasler, L. (2007). Blog search engines. *Online Information Review, 31*(4), 467–479.
Tsai, F. S. (2011). A tag-topic model for blog mining. *Expert Systems with Applications, 38*(5), 5330–5335.
Veksler, V. D., Govostes, R. Z., & Gray, W. D. (2008). Defining the dimensions of the human semantic space. In *Proceedings of the 30th annual meeting of the cognitive science society, Washington, DC USA, Cognitive Science Society* (pp. 1282–1287).
Wang, C., & Blei, D. M. (2013). Variational inference in nonconjugate models. *Journal of Machine Learning Research, 14*(1), 1005–1031.
Wang, X., & McCallum, A. (2006). Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, Philadelphia, PA, USA, ACM* (pp. 424–433).
Wang, Y., Agichtein, E., & Benzi, M. (2012). TM-LDA: efficient online modeling of latent topic transitions in social media. In *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining, Beijing, China, ACM* (pp. 123–131).
Wyner, A., & Engers, T. v. (2010). A framework for enriched, controlled on-line discussion forums for E-government policy-making. In *Proceedings of ongoing research and projects of IFIP eGOV and ePart 2010, Linz, PA Trauner Druck* (pp. 357–366).
Xu, C., Zhang, Y.-F., Zhu, G., Rui, Y., Lu, H., & Huang, Q. (2008). Using webcast text for semantic event detection in broadcast sports video. *IEEE Transactions on Multimedia, 10*(7), 1342–1355.
Yeh, J.-Y., Keb, H.-R., Yang, W.-P., & Meng, I.-H. (2005). Text summarization using a trainable summarizer and latent semantic analysis. *Information Processing and Management, 41*(1), 75–95.
Yuan, Q., Cong, G., Ma, Z., Sun, A., & Thalmann, N. M. (2013). Who, where, when and what: discover spatio-temporal topics for twitter users. In *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining, Chicago, Illinois USA ACM* (pp. 605–613).
Zhao, W. X., Jiang, J., Weng, J., He, J., Lim, E.-P., Yan, H., et al. (2011). Comparing twitter and traditional media using topic models. In *Proceedings of the 33rd European conference on advances in information retrieval, Dublin, Springer Press, Ireland* (pp. 338–349).
Zhang, F., Qiu, G., Bu, J., Qu, M., & Chen, C. (2009). Learning to Retrieve Opinions. In *Proceedings of the 10th Pacific Rim Conference on Multimedia: Advances in Multimedia Information Processing* (pp. 647–658). Bangkok, Thailand: Springer Press.
Zhu, L., Sun, A., & Choi, B. (2011). Detecting spam blogs from blog search results. *Information Processing and Management, 47*(2), 246–262.