8-2012

Community Interest as An Indicator for Ranking

Xiaozhong Liu
Syracuse University

Community Interest as An Indicator for Ranking

By
Xiaozhong Liu

xliu12@syr.edu
School of Information
Studies Syracuse University

Dissertation

Submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy in Information
Science and Technology in the Graduate School of Syracuse University

Advisor: Prof. Elizabeth Liddy

Committee members: Prof. Nancy McCracken
Prof. Jian Qin
Prof. Howard Turtle

Inside Reader: Prof. Bei Yu
Outside Reader: Prof. Jamie Callan
Committee Chair: Prof. Shui-Kai Chin

Acknowledgements

Doing research in information retrieval and natural language processing has always been interesting. Writing dissertation, however, is not always enjoyable. Thanks to my family, my wife Miao Chen, my parents and my lovely cat, who supported me a lot throughout this process. Their substantive help is a key reason I could accomplish this dissertation research.

I want to give my special appreciation to my knowledgeable advisor, Professor Elizabeth D. Liddy. She always took time for my research, however busy she might have been. During the past four years, she gave me incredible support and suggestions on my research. Meanwhile, she always encouraged me to find my own research topic, and so I can enjoy lots of academic freedom.

Professor Howard Turtle, my distinct committee member, gave me comprehensive help for my dissertation experiment. His expertise and experience in information retrieval contributed a lot to my dissertation experiment, from design to evaluation. His thoughtful comments and suggestions were really valuable for my dissertation and future research.

Professor Jian Qin is not only my academic committee member, but she also taught me what is research and how to become a real academic scholar. Her enthusiastic help was very important for my research.

I also would like to say thank you to my Yahoo! mentors Vadim von Brzeski and Reiner Kraft. I appreciate the great opportunity they offered to launch my preliminary experiment at their group.

At least, I want express my thanks to Professors Nancy McCracken, Bei Yu, and Jamie Callan. Your professional and thoughtful comments are important for my dissertation research.

Abstract

Ranking documents in response to users' information needs is a challenging task, due, in part, to the dynamic nature of users' interests with respect to a query. We hypothesize that the interests of a given user are similar to the interests of the broader community of which he or she is a part and propose an innovative method that uses social media to characterize the interests of the community and use this characterization to improve future rankings. By generating a community interest vector (CIV) and community interest language model (CILM) for a given query, we use community interest to alter the ranking score of individual documents retrieved by the query. The CIV or CILM is based on a continuously updated set of recent (daily or past few hours) user oriented text data. The interest based ranking method is evaluated by using Amazon Turk to against relevance based ranking and search engines' ranking results. Overall, the experiment result shows community interest is an effective indicator for dynamic ranking.

TABLE OF CONTENTS

Chapter 1: Statement of the Problem

1.1 Problem and Motivation

With the exponential growth of the web in the past decades, we are facing a flood of information. As a consequence, the increasing challenge of locating target information in cyberspace makes it important to design efficient and effective information retrieval systems.

Information retrieval (IR) is typically a two-step process. First, potentially relevant documents are identified through different retrieval models from collections, and then the retrieved documents are ranked by algorithm(s) based on different ranking mechanisms (Trotman, 2005).

In an ideal scenario, a user types a query in a traditional IR system, and he or she may either "find the target information" or "fail to find the target information" from the retrieved results. The two likelihoods can be expressed as {*search = success*} and {*search ≠success*}. Intuitively, the probability of search success may depend on the following three essential factors:

(1) *The likelihood that users can generate a high quality query to address their information need; e.g., given an information need, the probability that the user can generate a high quality query (query quality = high),*

$$\alpha: P(\text{query quality} = \text{high} \mid \text{information need}) \quad (1\text{-}1)$$

(2) *The likelihood that the system can present relevant results based on the (high quality) query; e.g., given a query, the probability that the system can find the high quality results (result quality = high),*

$$\beta: P(\text{result quality} = \text{high} \mid \text{query quality} = \text{high}) \quad (1\text{-}2)$$

(3) *The likelihood that users can find needed information in the (high quality) retrieved results; e.g., given retrieved results, the probability that the user can locate the needed information (search = success),*

$$\gamma: P(\text{search} = \text{success} \mid \text{result quality} = \text{high}) \quad (1\text{-}3)$$

If we assume α, β, γ are independent factors, the probability of achieving a successful search could be estimated by:

$$P(\{search = success\} \mid information\ need) \approx \alpha \cdot \beta \cdot \gamma \qquad (1\text{-}4)$$

While β is fully controlled by the system, α *and* γ are dependent on the user side, although the system may assist the user implicitly or explicitly to improve α *and* γ based on user or community knowledge. For example, in order to improve α, a system may use automatic query expansion or query recommendations to change the original query to better represent the user's information need; in the same way, other techniques, such as ranking, clustering or personalization, will change the format or ranking of the retrieved results and help users to better locate the target information, which will improve γ.

Most research in the information retrieval field can be classified into three types based on α β *and* γ. Some existing representative research topics are listed in the following table:

	Definition	**Controlled by**	**Example Research Topics**
α	Probability of a high quality query given an information need.	User, system	Feedback, context search, query expansion
β	Probability of high quality results given a query.	System	Retrieval models (e.g. vector space model, language model)
γ	Probability that the user can find needed information given retrieved results.	User, system	Ranking, result visualization, clustering, personalization

Table 1-1. Three essential factors affecting IR performance

For a given query, we could calculate the probability that *result quality = high* (β) by employing standard IR evaluation methods for an IR system. For instance, if we use *F1* score[1] (considering both precision and recall) as the judgment of the retrieved results, it can be defined:

Retrieved result quality = high …… $Score_{F1}(retrieved\ result) > k_1$

Retrieved result quality = medium …… $k_1 > Score_{F1}(retrieved\ result) > k_2$

Retrieved result quality = low …… $Score_{F1}(retrieved\ result) < k_2$

$$where\ Score_{F1} = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad and \quad k_2 < k_1$$

[1] Details will be reviewed in the second chapter

where k_1 and k_2 are the thresholds to categorize retrieved result quality.

Similarly, *precision-at-document-n* (Anh & Moffat, 2002) can be used as an indicator to estimate the probability that the user can find the needed information given retrieved (and ranked) results (*γ)*. That is, if top ranked documents are relevant (score of *precision-at-document-n* is high, e.g. larger than k_1'), the user is highly likely to find the needed information. Otherwise (e.g. *precision-at-document-n* is smaller than k_2'), *γ* will be low.

Compared with *β* and *γ*, in equation 1-4, *α* is the most difficult factor because the query is the only evidence to estimate the information need, and the "user information need" is almost immeasurable. Moreover, the qualities of queries across different "information needs" are incomparable. Belkin, Oddy, and Brooks (1982), for instance, suggested that the user is part of an IR system and they asserted that the representation of the anomalous state of knowledge (ASK) is the key component in understanding user's information needs.

If "user information need" is defined as a perfect or optimized query, *α* can be estimated by using the distance between $Query$ and $Query_{optimal}$. According to Gerard Salton and Buckley (1990) the optimal query could be defined as:

$$Query_{optimal} = \frac{1}{n}\sum_{rele-docs}\frac{D_i}{|D_i|} - \frac{1}{N-n}\sum_{Nonrele-docs}\frac{D_i}{|D_i|} \qquad (1\text{-}5)$$

where N is the total number of documents in the collection, *n* is the number of relevant documents (*N-n* is the total number of non-relevant documents) and |D| is the length of the document. If a user can input such an optimal query, it will maximize the effect of the relevant document collection while minimizing the non-relevant document collection. This conceptually optimized query is based on two conditions: first, formula 1-5 is a system-based (not user oriented) perfect query for a given collection (with *N* documents), and second, the relevant document sub-collection should be a known parameter (with *n* documents). The $Query_{optimal}$ will lead to the best retrieval performance given the detailed knowledge of the retrieval task and collection make-up. In terms of Salton's optimal query model, *α* can be estimated by:

$$\alpha \approx Distance(Query, Query_{optimal}) \qquad (1\text{-}6)$$

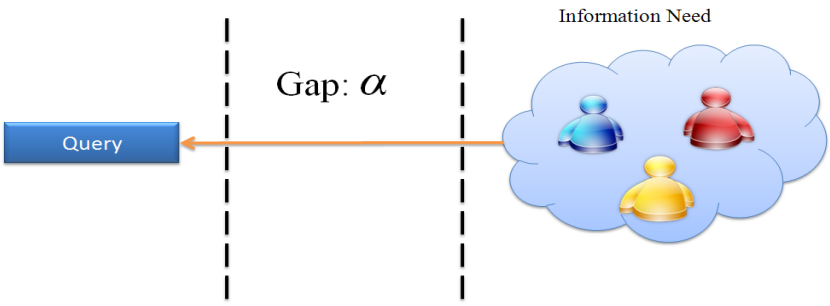

Figure 1-1. Distance between query and information need

Relevance feedback is the first retrieval and ranking mechanism that involved users and relevance judgments (relevant and non-relevant information) directly. However, identifying the entire relevant document sub-collection for $Query_{optimal}$ is an impossible task. Existing relevance feedback algorithms learn retrieval functions and additional query features or feature weightings from retrieved results, and typically require training data generated from relevance judgments by experts or users, which makes relevance feedback difficult and expensive to apply (Joachims, 2002). As a result of all of these factors, α inevitably becomes the most challenging component among these three factors.

From the classical IR perspective, a robust retrieval model could improve the probability of acquiring relevant results from the system side (improve β) while an effective ranking algorithm could help to improve the probability of locating the needed information from the user side (improve γ). As the following diagram shows:

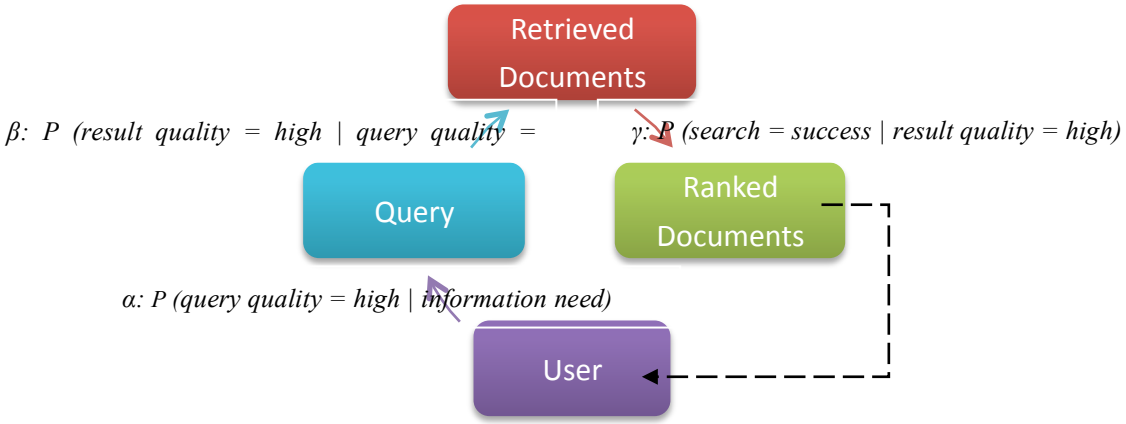

Figure 1-2. Three fundamental problems in classical IR

In practice, α and γ are closely related to each other. Intuitively, improving α, for example, by the

user's addition of informative query terms, helps the system to generate better ranked results and eventually improve γ. Ranking performance is highly dependent on the quality of the query, as most relevance-based ranking functions (e.g., vector space model) employ two parameters, as in the following equation:

$$Ranking\ Score\ (query, doc) \approx Relevance(doc, query) \qquad (1\text{-}7)$$

In equation 1-7, IR systems rank documents by their estimation of the relevance of candidate documents for a given user query by assigning numeric scores to each of them (such as a probability or similarity score) (Singhal, 2001), and relevance based methods integrate retrieval and ranking components (β *and* γ) together. Nevertheless, as mentioned earlier, this ranking function ignores the gap between the *user information need* and the *query* (α effect). Moreover, for most existing web search engines or general information retrieval systems, users tend to input short keyword queries instead of long ones (Jansen, Spink, Bateman, & Saracevic, 1998; Craig Silverstein, Marais, Henzinger, & Moricz, 1999), and when they do use longer queries, the query formulation process is not transparent to the users. In particular, without detailed knowledge of the retrieval task, collection make-up, and the retrieval environment, most users can hardly propose or update optimized query terms to satisfy the requirements of the ranking algorithms (Salton & Buckley, 1990), and this will result in a low quality query as well as low α and γ probability scores.

To address this problem, some more recent ranking algorithms employ statistical user centric data (to represent user or community preferences), such as (blog) citations, page hyperlinks, clickthrough and user search behavior data, to improve ranking performance. In these algorithms, the ranking score is partly assigned by the *popularity* of the results (retrieved documents) given statistical user data. And unlike the relevance based method, popularity based ranking (γ) is separate from the retrieval component (β).

$$Ranking\ Score(users,\ query) \approx Popularity(doc \mid stat.users\ data) \qquad (1\text{-}8)$$

As equation 1-8 shows, some new ranking algorithms use the popularity of retrieved documents to provide user oriented ranked results, namely *query-independent ranking models*. A successful case is the PageRank algorithm (Page, Brin, Motwani, & Winograd, 1998), which employs a web hyperlink structure as an indicator to "vote for" the popularity of each page. Similarly, click

through data, and other implicit or explicit user behavior data, are also widely used by search engines to compute the popularity of each page (Agichtein, Brill, & Dumais, 2006; Fox, Karnawat, Mydland, Dumais, & White, 2005; Joachims, 2002).

However, there are also some limitations to these algorithms. For instance, for a blog search engine, a blog posting getting a high number of citations or clicks (statistical votes), may be due to two different reasons:

1. *The content of the posting is relevant and interesting (it deserves a high rank), or*
2. *The blogger (author of the posting) is popular in a local community (the content may be pedestrian and does not deserve a high rank outside of this local community)*

The contexnt-free (statistical user data base) ranking algorithm will favor these postings no matter which scenario they belong to, and the ranking training data could be biased. Meanwhile, some statistical data on user behavior, such as click through data or dwell time, are only indirectly related to the target query.

Theoretically, choosing or integrating a popularity or relevance based ranking method is a challenging task. When α is low (the user cannot generate an accurate query), popularity based ranking is more effective because of its query-independent nature. Relevance based ranking makes more sense when α is relatively high (the user can propose a high quality query to address her information need, e.g., the query is similar to 1-5). *Query-dependent* and *query-independent ranking models* are like two sides of the same coin, and because α is an unpredictable and intangible factor, it's not easy to choose the most appropriate solution for each query automatically.

If we want to improve existing ranking algorithms as well as solve the dilemmas between relevance and popularity based ranking mechanisms, we could update the ranking function by integrating *user provided information*, along with the *query* and *documents,* for example by using relevance feedback (Rocchio, 1971), and focus on improving α and γ simultaneously, as shown in the ranking function 1-9:

$$Ranking\ Score \approx Ranking\ Function'(doc, query, user) \qquad (1\text{-}9)$$

Considering *user based information,* the *query* and *documents* the proposed ranking model has the following merits:

1. This ranking function (1-9) considers the gap between query and user information needs, which can improve α and solve the problem of relevance based ranking.

2. This is a *query-dependent ranking model*, which computes the ranking score in terms of query and solves the problem of popularity based ranking. (The problems associated with relevance based and popularity based ranking will be discussed further in the next chapter.)

Based on these assumptions, in this thesis, an exploratory investigation into how to use "*user interest*", or more generally, use "community interest" as an indicator to calculate the ranking score of the candidate retrieved documents is proposed. Because user or community interest may dynamically change over time, ranking, in this research, uses a computed measure of the interest level in the global (or local) community in a specific retrieved document for a given query at a given time.

$$Ranking\ Function(doc, query, user, time) \approx P_{interest}(doc|query, community) \qquad (1\text{-}10)$$

As equation 1-10 shows, the ranking score in this thesis is represented by the *current interest* score of the document for a given query and community.

Community interest is not the same as popularity ranking. There are three key differences between popularity and interest ranking. First, while most popularity algorithms are based on user generated statistical data, we view community interest as being represenTeded by the content (topics) of the candidate document as judged from a community perspective. Second, unlike most user behavior ranking algorithm, e.g. Agichtein et al. (2006), an interest based ranking model is query dependent. Last but not least, a community interest model is time-dependent as compared with the popularity model, which is based on past user behavior.

Obviously, the most challenging part in the ranking function is "*community*," which varies with the query and the retrieval system. Community interest is a dynamic variable, as community interest toward a query may change from time to time. In this thesis, we will use the chronological

user oriented textual data, specifically blog data, and various statistical models to estimate real-time community interest toward a query as well as evaluate its impact on ranking performance.

1.2 Contribution

From a system perspective, *community interest*, in this research, is defined as a dynamic topical probability distribution, which is trained from query dependent chronological user centric text data (like blogs) with the goal of improving γ. To date, there has been little research to study IR ranking from a dynamic community interest perspective. This thesis is expected to make both theoretical and practical contributions to information retrieval ranking methodology research, with particular contribution regarding real-time ranking.

At the theoretical level, this study aims to make two contributions:

First, from a user perspective, *community interest* is an innovative search context, which mirrors real-time user topical preference. Modeling community interest will help systems better understand a query and improve probability of α.

Second, this research will build the implicit relationships among *ranking*, *community interest* and *user generated text data* for real-time interest modeling as well as fill the gap between a query string and a user's information needs by ranking from a community interest point of view. Specifically, as the following diagram shows, this research demonstrates whether we can use user generated text data to estimate real time community interest by mathematical modeling and if computational community interest can be used as an indicator to improve ranking performance.

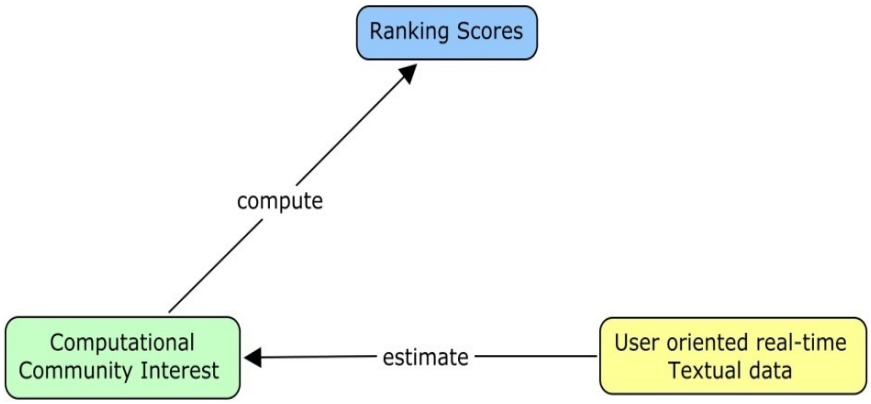

Figure 1-3. Relationships among ranking, community interest and user generated text data

In this thesis, as compared to traditional IR, the user oriented real-time text data, instead of the document collection, is used as the training collection for extracting the dynamic community interest.

At the practical level, this thesis aims to improve ranking performance compared with existing popular ranking algorithms, such as *vector space, BM25* and *language model*. Real world user evaluations are used to compare the new ranking algorithm to the ranking results from popular search engines, such as Yahoo or Google, which rely on, in part, complex statistical user behavior data. Standard information retrieval ranking evaluation methodology, *normalized discounted cumulative gain (NDCG),* is used in this thesis to analyze user judgments.

1.3 Research questions and goals of this thesis

This thesis is intended as exploratory work, investigating an innovative ranking method from a community interest perspective. More specifically, the goals are to:

1. *Develop the interest-based ranking mechanisms for real-time information retrieval systems.*
2. *Generate the real-time user oriented topical representation as a computational model for community interest, as well as use the real time interest model to rank or re-rank the retrieved results for a given query at a given time.*
3. *Evaluate the new ranking algorithm by comparing it with existing popular ranking algorithms and search engine ranking results.*

Meanwhile, the research questions are:

RQ1: What is community interest? And can we extract and model real time computational community interest from user textual data?

User or community interest has been studied by (Kim & Chan, 2008; Qiu & Cho, 2001; White, Bailey, & Chen, 2009) and these studies focus on personalized search as well as user profiling. In this thesis, the community is defined as a dynamic probability distribution over *topics* or *words* for each target query, and each topic or word interest probability is normalized by historical user

oriented text data.

RQ2: In what ways can real-time community interest be used to rank the retrieved results?

Ranking is the critical problem for information retrieval. In this thesis, we propose the innovative interest based ranking method by using community interest extraction (from RQ1). Two different ranking models are introduced in this research:

1. *Community interest vector ranking*, in that a community's interest is defined toward each query as a vector, and each component of the vector represents a (normalized) topic or word interest score related to the target query.

2. *Community interest language model ranking*. Based on classical language modeling, the retrieved documents are ranked based on the probability generated from the most recent community interest topic or word distribution, and this probability is then smoothed by historical community interest snapshots.

RQ3: How can we evaluate the real-time community interest ranking results? And can the community interest based ranking method improve rankings over existing methods.

The evaluation of a ranking algorithm is difficult, especially for the real-time ranking task, which cannot employ existing test collections such as TREC. *Precision-at-document-n* (Anh & Moffat, 2002) is currently a good measure for the web, as most users will be focusing on only the very first page of n results. *Normalized Discount Cumulative Gain (NDCG)* (Järvelin & Kekäläinen, 2002) works when user judged relevance data is available.

For this thesis, the most important contribution is to employ dynamic community interest as an innovative indicator for ranking. Since community interest may change from time to time, a real-world, real-time evaluation with users based on selected queries over a period of time was conducted. The questions for the user focused on "*Are you interested in this document based on the query right now?*" This question contains two meanings: 1. whether this document is relevant or not; and 2. whether user is interested in this document currently or not.

1.4 Limitation of thesis

There are two limitations to this research.

First, training data is critically important for dynamic community interest modeling (*RQ1*), but only a limited amount of data suitable for training is available for this research, and even with this limited data, some useful properties of the user data are inaccessible. This thesis uses blog data (a user oriented chronological text data set). Ideally, up-to-date blog data would provide the best data for training an interest model. However, existing large blog corpuses, such as the TREC blog dataset, represent are static and not up-to-date (when we launched the experiment). A compromise approach is to collect blog data from a blog search engine, but blog search engine may take some time (i.e. hours) to index the most recent blog postings thereby introducing a delay. In addition, the notion of "community" is variable, depending on the focus of a query, collection and the specific group of users searching. The community can be defined by any number of characteristic features such as geographic location, gender, occupation or hobby. Each community is likely to have different interests, corresponding to different interest topic distributions and different interest models. The blog data available does not, however, include such information as blogger, IP address, or location, and thus no analysis along these local community perspectives can be carried out. As a result, in this thesis, the studies will concentrate on the "global community", instead of a specific community, interest modeling for ranking. Work targeting a specific community is the subject of future work.

Second, the lack of comprehensive empirical evaluation (related to RQ3) is another major limitation for this thesis. Mentioned in section 1.1, overall, there are two different kinds of ranking algorithms, relevance based ranking and popularity based ranking. Most of the popularity based ranking algorithms are based on statistical user data, such as clickthrough data, dwell time and universal hyperlinks. This data is not available and without such data, it is difficult to implement those algorithms and use them as the baseline to judge the performance of the new ranking algorithm. The compromise is to compare the new ranking algorithm with existing web search engines' (such as Google and Yahoo) ranking results, an imperfect compromise, as these search engines' ranking mechanisms are usually a combination of different existing ranking algorithms,

proprietary and not open to inspection.

Chapter 2: Literature Review

2.1 Introduction

Information retrieval has been studied since the 1950s. According to Mooers' definition (Mooers, 1952), information retrieval is the name of the process or method by which a prospective user of information is able to convert his or her information need into an actual list of citations to documents in storage. Luhn (1957) was the first researcher to suggest using a statistical approach for retrieval and ranking tasks to address some of the existing retrieval problems. Since then, Luhn's statistical approach has proved to be the most successful method in this field. In more recent modern information retrieval studies, three basic components have been studied (Croft, 1993), as the model in Figure 2-1 shows.

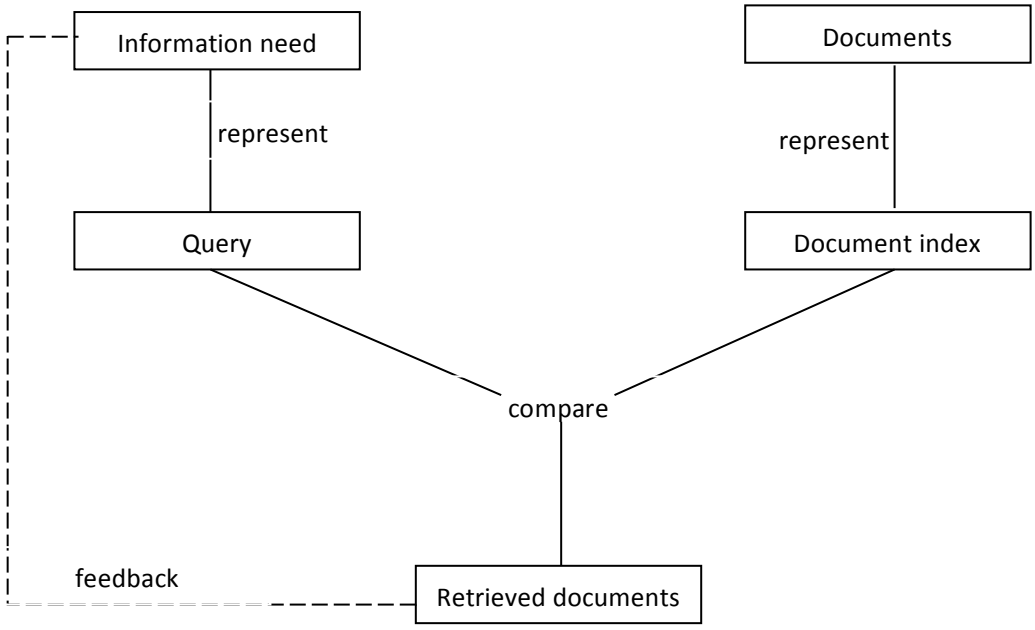

Figure 2-1. Information retrieval process

In this model, the user's initial information need is represented by a natural language statement, which is referred to as the query formulation process. Documents and collections are indexed by a list of features, such as bag-of-words, which assumes that the semantics within a text can be represented by an unordered collection of words. Last, the query is compared against the document representation resulting in a ranked candidate relevant document list.

For most web retrieval systems, a very large number of documents are indexed, and the ranking

model is the central problem. An effective ranking algorithm can position the most relevant documents somewhere at the top of the ranking list, which will reduce the time users invest to find the needed information. For some methods, such as classical vector space (Salton, Wong, & Yang, 1974) or language model (Ponte & Croft, 1998b), retrieval and ranking are the same step, while some other algorithms, such as PageRank (Page, et al., 1998) or HITS (Kleinberg, 1999), employ a two-step process. First, potentially relevant documents are identified through retrieval algorithms, and then the retrieved documents are ranked with a ranking algorithm by estimating their degree or probability of importance to the user. In this thesis, an innovative ranking method is explored and community interest is used as the new indicator for ranking the retrieved documents for a given query at a given time.

Chapter 2 consists of three main parts that review the theoretical and practical research methodologies in the information retrieval and text mining fields that are the background of this research. In sections 2.2, 2.3 and 2.4, some of the most influential models, theories and algorithms will be reviewed targeting the three main challenges (α, β and γ) mentioned in the first chapter. In section 2.5, topic extraction and topical representation related methods, used to represent community interest in this thesis, will be covered. Lastly, evaluation methodologies, especially web based ranking related evaluation, will be reviewed.

2.2 Information need and query (α)

As stated in the first chapter, the unpredictable gap between the user information need and the user generated natural language query is the first challenge for information retrieval. The probability based gap between the information need and query can be defined as follows:

> *The likelihood that users can generate a high quality query to address their information need; e.g., given the information need, the probability that users can generate a high quality query (query quality = high),*

$$\alpha\text{: } P(\text{query quality} = \text{high} \mid \text{information need}) \qquad (2\text{-}1)$$

The definition of α enables us to characterize the distance between query and information need. However, first, the distance between query and information need is almost immeasurable, as the information need is not a measurable variable. Second, in most cases, α is smaller than 1 and, for

most users, without knowledge of the collection, relevant documents and the retrieval task, α can be a very small number. Last, even with an initial query formulation process on the user side, researchers in information retrieval have found a number of techniques to improve α automatically or semi-automatically. In this section, some of the most influential works that focus on improving α will be reviewed.

2.2.1 User information need and query formulation in information retrieval

User query formulation is known to be a hard problem. For most ad-hoc retrieval system users, the query formulation process may result in a vast amount of information, and lead to an overwhelming feeling of being lost in conceptual space (Hofstede, Proper, & van der Weide, 1996). In the web search environment, very short queries are challenging existing retrieval and ranking methodologies, given the truth that query length in information retrieval systems is well known to be positively related to effectiveness of retrieve results (Belkin, et al., 2003). Existing attempts on increasing query length mainly focusing on automatic pseudo feedback, while some researches increase query length by encouraging users to input additional query terms (Karlgren & Franzén, 1997; Belkin, et al., 2002), which will be mentioned in the next section.

From retrieval perspective, query formulation is closely related to retrieval or ranking performance. Cronen-Townsend, Zhou & Croft (2002) developed the method to predict query performance by using relative entropy between query and collection language models. The experiment on TREC collection achieved positive result. In the web search environment, collection is more heterogeneous than other experimental text collection. The similar query prediction algorithm applied on GOV2 (web collection) shows that it is different to predict query performance in a large web collection (Carmel, et al 2006). Another similar experiment by Zhou & Croft (2007) first divided web query into content-based and named-page finding groups, and then different prediction methods were applied to those query categories, which resulted in better precision.

Some researchers have studied the user query formulation process (given the information need) from the user behavior perspective (Lau & Horvitz, 1999; Maglio & Barrett, 1997; Marchionini,

1997; Thomas & Fischer, 1996). For instance, Lau and Horvitz (1999) analyzed one day's web query log, and carried out an experiment to study how users refined their own queries. Users were identified by the similarities in time and topic of an information need and identified by a GUID (globally unique identifier). They found that a user's initial queries poorly represented their information needs. Based on user changes to the original query, they identified six different query refinement classes:

- New: A query for a topic not previously searched for by this user within the scope of the dataset.
- Generalization: A query on the same topic as the previous query, but seeking more general information than the previous query.
- Specialization: A query on the same topic as the previous query, but seeking more specific information than the previous query.
- Reformulation: A query on the same topic that can be viewed as neither a generalization nor a specialization, but a reformulation of the prior query.
- Interruption: A query on a topic searched on earlier by a user that has been interrupted by a search on another topic.
- Request for Additional Results: A request for another set of results on the same query from the search service. Duplicate queries appear in the data when a person requests another set of results for the query.

By studying users' self-refinement of queries, we know users normally input the first search action based on their search goal, which, sometimes, does not meet the system's requirements or satisfy their own goals. After a certain time interval, they are likely to re-send a follow up search action based on the same search goal, and they could learn from the initial search action. Similar query formulation studies were run by Maglio & Barrett (1997). Given a low quality query, query refinement and relevance feedback are frequently used to automatically assist users to accomplish the abovementioned steps by learning the possible relevant knowledge from a query or retrieved result perspective.

Query formulation studies have also focused on mining and analyzing a large quantity of query logs (Beitzel, Jensen, Chowdhury, Grossman, & Frieder, 2004; Jansen, et al., 1998; Silverstein,

Henzinger, Marais, & Moricz, 1998; Silverstein, et al., 1999). Large scale query log analysis results show in web search engines, users tend to input very short queries. For instance Silverstein, et al. (1999) showed that 87.4% of the user typed queries used three or fewer terms, with an average number of 2.35 terms per query. Query length is an important indicator of the quality of a query, Anh and Moffat (2002) experimented with forty-nine TREC queries of two to six terms. Their findings show that the probability of finding matching results increases dramatically as the length of the query increases and concluded that short queries do not work well within the traditional retrieval models, such as vector space framework.

Second, most users using web search engines are not expert searchers and they have limited knowledge for characterizing their information need. For example, users seldom use advanced query features such as utilizing Boolean operators in their queries.

Third, the distribution of queries and query terms are skewed. Users are interested in certain queries and query terms more than others, and query repetition is a common phenomenon. Moreover, within a specified period of time, such as an hour or a day, a users' query distribution over specific topics is quite stable.

The first two findings challenge the existing retrieval and ranking algorithms, because of the large gap between query and information need and the small value of α. The last finding tells us that users' interest over some queries and topics can be stable and it is thus possible to employ an up-to-date and query-dependent community interest model as an indicator to improve ranking performance as well as to shorten the distance between query and information need.

Integrating user and community preferences into retrieval and ranking models to fill the gap α has been studied since the 1970s. Generally, extracted computational user preference or community preference is used for three different but related purposes in the existing research:

1. Improving query quality by leveraging user relevance or non-relevance judgments, which helps the system better understand a user's information need. An example of this is relevance feedback (Rocchio, 1971; Salton & Buckley, 1990).

2. Updating the original user query or changing the weight of the existing query terms from query logs or documents, as in the work of (Baeza-Yates, Hurtado, & Mendoza, 2004b; Kwok,

1996).

3. Ranking or re-ranking the retrieved results based on user, community or global preferences, which are extracted from different search related resources. Examples of this are linkage based (Page, et al., 1998), click based (Joachims, 2002), behavior based (Agichtein, et al., 2006) and context based (Kraft, Chang, Maghoul, & Kumar, 2006) ranking methodologies.

2.2.2 Feedback

Relevance feedback, introduced by Rocchio (1971), was the first retrieval and ranking mechanism that effectively involved users and relevance judgments directly. In most retrieval systems, the query formulation, or reformulation process is a user controlled, manual process (Salton & Buckley, 1990), and it was found that users have difficulty entering high quality query terms to satisfy the existing retrieval and ranking algorithms. For instance, in order to formulate a high quality query, users need to predict terms appearing in the relevant document. In addition, they have to avoid using ambiguous terms that may appear in irrelevant documents, but such tasks may be very difficult for most users (Vélez, Weiss, Sheldon, & Gifford, 1997). To assist, we can utilize a user's direct relevance or non-relevance judgments to modify their initial queries to increase probability α.

Relevance feedback, introduced in the mid 1960s, is a controlled and automatic process for query reformulation, and it has been proven an easy and effective method (Rocchio, 1971; Salton & Buckley, 1990). The basic process of relevance feedback is shown as follows. First, if the initial query is a term based vector:

$$Q_{initial} = (q1, q2 \ldots qn) \tag{2-2}$$

where q1 to qn is the weight of each term in the query, then the document collection can be classified into relevant and non-relevant groups. The optimal query (leading to the best match where $\alpha \approx 1$) could be (Salton & Buckley, 1990):

$$Q_{optimal} = \frac{1}{n} \cdot \sum_{\mathrm{Re}levant-Docs} \frac{D_i}{|D_i|} - \frac{1}{N-n} \cdot \sum_{Nonrelevant-Docs} \frac{D_i}{|D_i|} \tag{2-3}$$

where N is the number of total documents in the collection, n is the number of relevant documents

(N - n is the total number of non-relevant documents) and |D| is the length of the document. If the user can input an optimal query, it will maximize the effect of the relevant document collection while minimizing the non-relevant document collection in the retrieved results. However, we know that n is an unknown parameter as we cannot identify the relevant document collection based on the initial query. If there is an existing subset of relevant documents (n_1), and another set of non-relevant documents (n_2), we can write a new query to estimate the optimal query based on existing information (Rocchio, 1971):

$$Q_{new} = Q_{initial} + \frac{1}{n1} \cdot \sum_{\substack{Known \\ \text{Re}levant}} \frac{D_i}{|D_i|} - \frac{1}{n2} \cdot \sum_{\substack{Known \\ Nonrelevant}} \frac{D_i}{|D_i|} \qquad (2\text{-}4)$$

For a subset of documents judged by the user, the weight of the relevant document vector will positively affect the query vector; similarly, the non-relevant document vector will negatively affect the initial query. In this way, the probability of α. will be improved by leveraging the user's relevance judgment.

In real world systems, relevance feedback is always desirable to improve probability α; however, users may not want to provide explicit relevance judgments on the initial retrieval result. The compromise is a pseudo feedback method. The basic idea is to assume a small number of top-ranked documents from the initial retrieved result to be relevant documents, and use them to expand or update the weight of the initial query (Tao & Zhai, 2006; Xu & Croft, 2000). The process of pseudo feedback is shown in Figure 2-2. While existing relevance feedback algorithms learn retrieval features and functions from retrieved results, they typically require training data generated from relevance judgments by experts and users, which makes relevance feedback difficult and expensive to apply. Only a small percentage of users would likely provide the explicit feedback (Joachims, 2002). Compared with that, pseudo feedback, which assumes that the top k documents in the retrieved results are relevant to the original query, and uses them to change the query for ranking, is easier and cheaper to use.

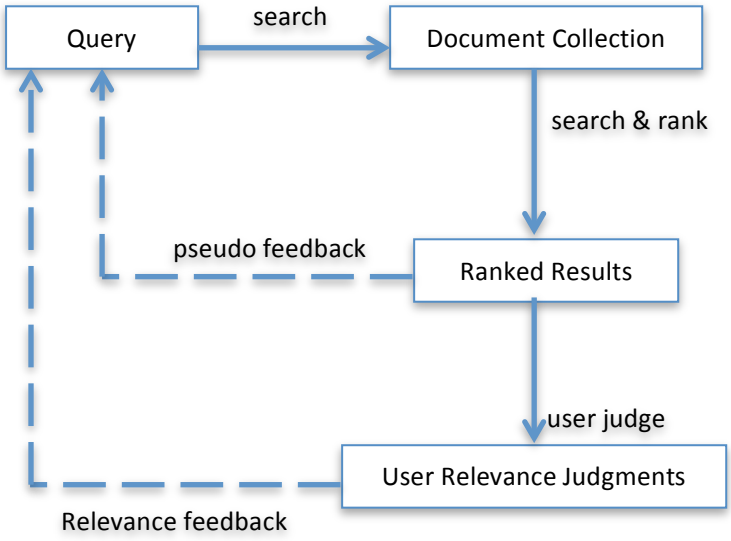

Figure 2-2. Relevance feedback and Pseudo feedback

The drawback for fully automatic relevance feedback is that some added terms may have different meanings from the user's intended meaning (the polysemy effect), which leads to a degradation of precision (Sparck Jones, 1972). So, each time the expansion algorithm brings a new term to the query, we have to take the risk of negatively affecting the original query vector.

An alternative feedback method is the probabilistic feedback model first introduced by (Robertson, Rijsbergen, & Porter, 1980). The difference is that it applies a probabilistic model to create the relevance model, and it calculates the ratio of the relevant probability and non-relevant probability of each candidate feedback term, as in the following ranking method:

$$log \frac{P(term|relevant)}{P(term|non-relevant)} \qquad (2\text{-}5)$$

A language model (the details of which will be reviewed in the next section) can also be used for feedback. From the classical language model perspective, the query is regarded as a sample of a query language model. It is hard to interpret query expansion or modification by adding additional terms or adjusting the initial query (Zhai, 2008). To address this problem, the Kullback-Leibler (KL) divergence retrieval model measures the distance of the document language model and the query language model (Lafferty & Zhai, 2001). In this KL model, this distance defines the score of a document D with respect to a query Q by:

$$S(D,Q) = -D(\theta_Q||\theta_D) = \sum_{w\in V} P(w|\theta_Q)\log P(w|\theta_D) - \sum_{w\in V} P(w|\theta_Q)\log P(w|\theta_Q) \quad (2\text{-}6)$$

So, the remaining problem is to estimate θ_Q and θ_D. As mentioned earlier, θ_D can be estimated from a traditional language model. However, θ_Q estimation offers the opportunity for relevance feedback. For instance, for given feedback documents collection F, the new $\theta_Q^{'}$ could be:

$$P\left(w\middle|\theta_Q^{'}\right) = (1-\alpha)P(w|\theta_Q) + \alpha P(w|\theta_F) \quad (2\text{-}7)$$

Modeling θ_F is another challenging task, and the words in feedback documents F include two kinds of words: background words and topical words. While the first part can be explained by *P(w|C)*, the second part could be interpreted as the target, $P(w|\theta_F)$. In order to discriminate the topical words (by assigning higher probability score) from others. (Zhai & Lafferty, 2001) proposed two methods to estimate θ_F. For the first method, the log likelihood function is:

$$\log P(F|\theta_F) = \sum_{w\in V} c(w,F)\log((1-\lambda)\,\mathrm{P}(\mathrm{w}|\theta_\mathrm{F}) + \lambda\mathrm{P}(\mathrm{w}|\mathrm{C})) \quad (2-8)$$

λ, from 0 to 1, is the parameter that controls the contribution of background terms (smoothing), and 1- λ is the contribution of $P(w|\theta_F)$ to the function value. The other method employed is the Expectation-Maximization (EM) algorithm, which involves the KL-divergence between the language models of background, θ_C, and each individual document in the F, θ_i.

Meanwhile, other language model based feedback algorithms have been used for optimizing smoothing parameters or query term re-weighting, for example Croft, Cronen-Townsend, & Larvrenko (2001), Hiemstra (2001), *Ng* (2000), and Shen & Zhai (2005).

Another related research is query expansion from document the side. One of the earliest studies in query expansion is the work of Spärck Jones (1971), who used word clusters extracted from documents to expand an initial query. More recently, Qiu and Frei (1993) studied query expansion by using concepts extracted from the document collection. Similar research can be found in Deerwester, Dumais, Furnas, Landauer, & Harshman (1990). Kwok (1996) and Xu and Croft (1996) compared the retrieval performance of local document based query expansion (e.g., pseudo feedback) and global document based query expansion (e.g., query expansion from an entire document collection). Based on the TREC experiments, they found that performance using local

document analysis on top ranked retrieved documents can be better than global document analysis (that is, on the corpus).

In this thesis, a key step is to estimate community interest from a collection of user generated (real-time) text data. As for documents, there are also two kinds of information in the collection: user interest topic words and background topic words; meanwhile, the ranking algorithm also needs to discriminate the current user topic words from historical user topic words. In section 3, we will propose a method to estimate the community interest language model based on the work of Zhai & Lafferty (2001). Community interest based ranking can be viewed as a type of user interest based automatic feedback.

2.2.3 Automatic query optimization

In contrast to relevance feedback, query optimization, such as query suggestion, query expansion or query term re-weighting, is frequently used by web search engine systems to modify original user queries by looking at query logs or user sessions instead of looking at documents. A central problem of query suggestion is how to model the information needs associated with a query, given existing knowledge (Baeza-Yates, et al., 2004b). Basically, there are two different methods: get the related terms from user query logs, or get the related terms from the search results or collection.

For a query log based approach, the basic logic of query suggestion is to calculate the similarities between different queries. If two queries are close enough, then there is high probability that they can provide term suggestions to each other that may serve the same information need. Calculating query similarity is the central problem. (Wen, Nie, & Zhang, 2001) suggested that query similarity can be defined as 1) keywords in the query, 2) a string match of whole queries, 3) common clicked URLs, and 4) the distance of the clicked documents in the result collection. For example, D(p) and D(q) are the document collections presented to the user by query q and p, and D_C(p) and D_C(q) are the document collections selected by user. If $D\_C(p) \cap D\_C(q) \neq \ominus$, then the shared documents represent the similarity between q and p:

$$Similarity_{cross\_doc}(p,q) = \frac{RD(p,q)}{Max(rd(p),rd(q))} \qquad (2\text{-}9)$$

where rd( ) is the number of clicked documents for a query, and *RD(p, q)* is the number of document clicks in common. Similarly, Zhao et al. (2006) used query session and calculated similarity between two queries by measuring their popularity over time. Fonseca, Golgher, Moura, and Ziviani (2003) used association rules to discover the relatedness and similarity between queries by using query logs and query sessions. However, these methods (based on user sessions) are only useful when the target queries are popular, requiring a great number of clicks. In order to remedy this drawback, Zhao et al., (2006) computed the similarities by comparing two queries' results ranking. The assumption is that if the queries are similar, their resulting documents' rankings should be similar.

Since (short) queries in ad-hoc retrieval systems do not adequately represent the user information need and are usually "flat", that is, without frequency information to help differentiate important terms from others (Kwok, 1996), query term re-weighting and query expansion are used to update initial queries by using evidence from the document side.

Kwok (1996) deduced query term importance weighting from the document collection using different methods, such as average within-document term frequency and inverse log document frequency with cut-off and "peaking" adjustments. The experimental results on TREC data showed that the re-weighted query could significantly improve retrieval performance.

In other research, a domain specific knowledge based thesaurus is used to expand queries in specialized retrieval systems. Aronson and Rindflesch (1997), Hersh, Price, and Donohoe (2000) used UMLS in the medical domain for query expansion. But this ontology base query expansion technique is not popular in general domain search engines, which must support queries in various and limitless topics.

2.3 Information retrieval model (β)

The retrieval model is the core of information retrieval from a systems point of view. Most retrieval models work solely on the system side and they calculate the degree or probability that a document is relevant to a given query. Based on the definition in the first chapter, *β* is:

The likelihood that the system can present relevant results based on the (high quality) query; e.g.

given query, the probability that the system can find the high quality results (result quality = high),

$$\beta: P(\text{result quality} = \text{high} \mid \text{query}) \qquad (2\text{-}10)$$

In this section, we will first briefly review different types of retrieval models and then focus on two kinds of models, which are used in this thesis: the vector space model (Salton, et al., 1974) and the language model (Zhai & Lafferty, 2001).

2.3.1 Retrieval models

The basic question for retrieval is: given a query *q*, how can we know if a document *d* is relevant to *q*? If we integrate the ranking part; the question can be changed into: given a query q, how can we know if a document *d1* is more relevant to *q* when compared with document *d2*?

There are three basic approaches for retrieval.

Similarity approach: this approach is used to find the similarity between the query and the retrieved document to represent the degree of relevance. Examples are the vector space model (Salton, et al., 1974) and the probability distribution model (Wong & Yao, 1989). The basic assumption is that if a document uses more query terms (thus more similar to the query), the document is more likely to be relevant to the target query. In this approach, the document and query are represented in the same way and they are key in deciding how to define the similarity function.

Probability approach: this approach computes the probability of relevance given a document and a query *P(relevant | doc, query)*. The generative model is the main method for this approach. Examples are the classical probability model (Robertson & Jones, 1976), which computes document generation probability given query *P(doc | query)*, the language model (Ponte & Croft, 1998a), which calculates the query generation probability *P(query | doc),* and BM25 (Robertson, 1997), which ranked the document by the log-odds of their relevance.

Probabilistic inference approach: in this approach, the retrieval task is stated as an inference or evidential reasoning process. For instance, Turtle and Croft (1991) proposed the inference network model. The documents and queries are represented as a hierarchical reference network, while the

information need is the root node in the network.

In this section, two major retrieval models, the vector space model and the language model, will be reviewed. They are used in this thesis as methods to extract the dynamic community interest model for ranking. The third approach, inference network, can be combined with language model into a single framework (Metzler and Croft 2004). As a result, if we prove that the language model can be successfully used to extract community interest then the inference network approach may also potentially be useful in solving this problem.

2.3.2 Vector space model

Following Luhn's retrieval criterion (Luhn, 1957), the words appearing in both queries and documents are the key feature in designing a retrieval system. The remaining question is: how can we know if one document is more relevant than another? Intuitively, if document A contains more query terms, or more important query terms, than B does, then A is more likely to be judged relevant than B, and the ranking score is based on the estimated likelihood of relevance, as the following formula shows:

$$Ranking\ Score(doc, query) \approx Relevance(doc, query) \qquad (2\text{-}11)$$

Based on this hypothesis, Salton, et al. (1974) proposed a vector space model (VSM), which indexes documents and queries as vectors and each component is associated with a term's weight in the document or query:

$$\overrightarrow{doc} = (w_{d1}, w_{d2}, \dots \dots w_{dn})$$

$$\overrightarrow{query} = (w_{q1}, w_{q2}, \dots \dots w_{qm}) \qquad (2\text{-}12)$$

The degree of relevance in VSM can be calculated by the cosine similarity between query and document vectors:

$$Relevance(\overrightarrow{query}, \overrightarrow{doc}) = \frac{\sum w_{dj} \cdot w_{qj}}{\sqrt{\sum (w_{dj})^2 \cdot \sum (w_{dj})^2}} \qquad (2\text{-}13)$$

The remaining problem is how to weight each term in the vector to optimize the ranking

performance. There are two reasons for term weighting in information retrieval and ranking: 1) terms likely to be relevant to the user's need should be retrieved, and 2) terms likely to be extraneous should be rejected or penalized (Salton & Buckley, 1988). Term frequency (TF) weighting was used in the earliest retrieval systems, which resulted in good recall, but very low precision, because most high frequency terms are not only concentrated in few documents, but the whole collection. The well-known solution is to use inverse document frequency (IDF) (Spärck Jones, 1972). Compared with various TF functions, the definition of IDF is quite uniform, as the following formula shows:

$$IDF(t) = \log\left(\frac{N}{k}\right) \qquad (2\text{-}14)$$

where N is the number of total documents in the collection and k is the number of documents containing term t. The weight of the term is then calculated by *TF*IDF*. For a specific term, if it appears in almost all the documents in the collection, $\log\left(\frac{N}{k}\right)$ will be almost 0 (as $N \approx k$), and the weight of this term will be almost 0. IDF is a very effective method to filter terms that do not discriminate documents in the collection for ranking and retrieval. In the Language Model, smoothing has the same effect from the probability perspective (Zhai, 2008), and it models the background language model by using $P(term|C)$.

IDF effect based noisy modeling is a very important concept for this thesis. Community interest in this research is represented by a topic distribution that represents the current probability of interests. However, there are two kinds of topics (from a topic extraction algorithm), community interest topics and background topics. Community interest topics are those a community is likely to be interested in, such as the breaking news about the query topic. But a community's interest toward background topics could be stable. From an IDF perspective, if user's information need is quite dynamic (i.e. a news event's recent change), these background topics should be identified and penalized by the ranking algorithm.

The vector space model is one of the most successful retrieval models. According to (Baeza-Yates & Ribeiro-Neto, 1999), even detailed information retrieval ranking algorithms used by major search engines are not publicly available, however, it seems that most use term weighting or variations of vector space models.

One of the major limitations of vector space models is on the query side, where the users' query terms may fall short of representing their information need (result in a low α). Anh and Moffat (2002) said that "Users of web search engines are notoriously parsimonious in their use of search terms, and search effectiveness has tended to be relatively poor on the resulting short queries, especially when compared against the good performance attained by recent systems when working with long TREC-Iike queries." As a result, we need better techniques to rank the documents, given an unpredicted query.

2.3.3 Language model

Unlike vector space, the language model calculates the probability of relevance of a document *doc* with respect to a query q by estimating the likelihood of generating q from *doc.* This was first introduced by Ponte and Croft (1998a), and can be described as the following formula:

$$Relevance(doc,\ q) \approx P(q \mid doc) \qquad (2\text{-}15)$$

In the past decade, a remarkably large number of publications cite cases in which statistical language models are used to compute the ranking scores of retrieved documents in a given query. The experimental results show that language models are effective and robust retrieval models, and that the use of language models can be flexible.

The language model based retrieval and ranking function is described by Zhai (2008). Given each document in the collection, a two-step statistical model defines the probability of generating the user query. Documents are then ranked according to this probability. When a query is entered, the system first uses the query formulation model to hypothesize the terms that might be generated for each word in the request. This results in a structured query that represents all queries that might have generated the request. In a second step, the system uses the matching model of each document to calculate the probability that the document generated any of the queries represented by the structured query.

A major challenge for language models is how to accurately estimate the document language model. The most straightforward method is to use the document as direct evidence for estimation. However, a single document is only a small sample, and usually a smoothed estimate of the

document models based on the background collection term frequencies is used to adjust the document language model.

Hiemstra (1998) introduced ranking based on a mixture of global and local probability distributions that are the logic of language model smoothing. Smoothing is one of the most important components of the language model, which refers to the adjustment of the maximum likelihood estimator of a language model to make the query generation probability more accurate. Smoothing has two different functions (Zhai, 2008):

- First, it addresses the data sparseness problem. As a document is only a very small sample, the probability *P (qi | Doc)* could be zero for those unseen words (Zhai & Lafferty, 2004).
- Second, smoothing helps to model the background (non-discriminative) words in the query.

Compared with TF-IDF weighting parameters, the smoothing parameter is more meaningful from the point of view of statistical estimation (Zhai, 2008).

Miller, Leek, and Schwartz (1999) use hidden Markov models for ranking, including the use of bi-grams to model two word phrases and a method for performing blind feedback. Song and Croft (1999) used a model which includes bi-grams and introduced Good Turing re-estimation to smooth the document models.

2.4 Ranking methodology (γ)

Ideally, an optimized information retrieval system should present the important documents high in the ranking results, with less important documents following below to minimize the time users invest in interpreting the retrieved results. However, it is necessary to first define what an "important" retrieval result is.

> *The likelihood that users can find needed information from the (high quality) retrieved results; e.g. given retrieved results, the probability that a user can locate the needed information (search = success)*
>
> $$\gamma: P(search = success \mid result\ quality = high) \qquad (2\text{-}16)$$

In web information retrieval systems, exponential growth of the number of documents makes

ranking the most important component. Ranking will directly help users to access their needed information effectively, and it plays the decisive role in predicting probability γ in the web search environment. If users' needed information (most relevant document) is on the top of the ranked list, users are highly likely to find the information. If not, the search task is more likely to fail. According to the (Jansen, et al., 1998) experiment on query sessions (for general web search), analysis shows that most users, 58%, did not access any results beyond the first page; this is likely representative of search behavior and does not necessarily mean users are satisfied with the top ranked retrieved results.

2.4.1 Popularity based ranking

As shown in the last section, in classical IR, relevance based computation between a query and a retrieved document, such as similarity based relevance (Salton, et al., 1974), probability based relevance (Robertson, 1977) and language model based relevance (Ponte & Croft, 1998a) is used to rank the result collection. In these ranking mechanisms, the ranking and retrieval modules could be one and the same, namely, they both identify the candidate documents to be retrieved and score the degree or probability of relevance for each of them.

In web retrieval systems, where documents are rich in links, tags and user behavior data, there is increasing use of ranking algorithms that rank the documents based on their own importance. The most well-known algorithm is PageRank (Page, et al., 1998), which employs the hyperlink structure of the web pages for ranking. Clickthrough or other implicit user behavior data are also frequently used as the ranking parameters by current search engines (Agichtein, et al., 2006; Fox, et al., 2005; Joachims, 2002).

In this section, some well-established popularity based ranking methods will be reviewed.

The web is an example of a social network. Social network analysis has been extensively researched since the 1950s, long before the advent of the web (Scott, 1988). Social network analysis is concerned with properties related to connectivity and distances in graphs, with diverse applications like epidemiology, espionage, citation indexing, and so on (Chakrabarti, 1999).

There are two well established algorithms based on web linkage and social networks: PageRank,

and HITS. PageRank is the trademark ranking algorithm of Google (Page, et al., 1998). Generally speaking, PageRank relies on the uniquely democratic nature of the web by using its vast link structure as an indicator of an individual page's value; it is a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank interprets a link from page A to page B as a vote (by page A, for page B), and for each page (node) on the network, there will be incoming links and outgoing links, which are the parameters of page importance.

$$PR(A) = (1-d) + d \cdot (\frac{PR(T1)}{C(T1)} + \frac{PR(T2)}{C(T2)} + ... + \frac{PR(Tn)}{C(Tn)}) \qquad (2\text{-}17)$$

Above is the formula of the PageRank algorithm. For the entire web page (or document) collection indexed in the database, assuming that there are *n* documents that have hyperlinks (or citations) pointing to page *A, as T1 ⋯ Tn*, and the count of link in each page is *C(T1) ⋯ C(Tn)*, then recursively, each page that links to page A will vote for the popularity of A. The vote score will be normalized by the number of links on the page.

The HITS (Hypertext Induced Topic Selection) algorithm (Kleinberg, 1999) is another popularity based algorithm for ranking, which can replace the PageRank algorithm in a web search engine for ranking purposes. For the HITS algorithm, each page has two different popularity scores: Authority and Hub, which possesses mutual recursion. The underlying assumption is that a good authority is pointed to by many good hubs and a good hub points to many good authorities.

From the above two ranking algorithms, it has been found that hyperlink based social network structure is a significant characteristic in the web environment, and calculating popularity score based on networks can effectively improve the ranking performance for a web information retrieval system. Other related web media can also get benefits from these kinds of ranking algorithms.

For instance, a blog is an extension medium of the traditional web (namely, blogs inherit all the properties of the web), and the relatedness between blogs are not just limited to hyperlink based citation. Instead, much richer types of links or relationships exist, like the blogroll (relatedness between blogs), permalink (a pointer or reference ID for the specific posting), comments,

trackback (an automatic communication that occurs when one weblog references another), and cross-media (reference of other user-oriented media outside blogosphere). The Blog track was introduced in 2006 as part of the TREC evaluation corpus, for instance Blog track in TREC 2006 (Ounis et al. 2007) and TREC 2007 (Macdonald, Ounis, & Soboroff 2008). In Blog track, traditional relevance judgments were provided, and innovative tasks were provided such like sentiment analysis, which are based on new web elements of blog (i.e. comments or permalink). As a blog is employed in this research, more investigation is needed in this area for social network based blog retrieval and ranking.

Since the blogosphere is a sub-class of cyberspace, most web-structure based ranking algorithms can be directly applied to blog ranking, such as PageRank and HITS. Some recent studies have used updated web algorithms to improve their performance in the blog context. For instance, Fujimura, Inoue, and Sugisaki (2005) proposed the EigenRumor algorithm in the blog context. The EigenRumor design is based on the logic of the PageRank and HITS algorithms, which deal with hub score and authority, but also add a new reputation score to better satisfy the characteristics of blogospace.

In the EighnRumor algorithm, each blogger is an information provider, namely an agent, while each blog posting is an information object. Two kinds of relationships exist between agent and object: information provisioning (agents write postings) and information evaluation (agents write comments). Extending the HITS algorithm, EighnRumor defines three different popularity scores:

- Authority score (agent property). This indicates to what level agent i provided objects (postings) in the past that agreed with the community direction. The higher the score, the better the ability of the agent to provide objects to the community.
- Hub score (agent property). This indicates to what level agent i submitted comments (evaluation) that agreed with the community direction regarding other past objects. Similar to authority, the higher the score, the better the ability of the agent to contribute evaluations to the community.
- Reputation score (object property). This indicates the level of support object j received from the agents, i.e., the degree to which j agrees with the community direction. Again, the higher

the score, the better the object conforms to the community direction.

As a result, the algorithm provides us three different vectors instead of two vectors like HITS. These vectors tell us how a blogger contributes to the posting as well as how a posting gets credits from the blogger. Similar to PageRank and HITS algorithms, EighnRumor has the following assumptions underlying computation of the popularity scores:

> Assumption 1: The objects that are provided by a "good" authority will follow the direction of the community.
>
> Assumption 2: The objects that are supported by a "good" hub will follow the direction of the community.
>
> Assumption 3: The agents that provide objects that follow the community direction are "good" authorities of the community.
>
> Assumption 4: The agents that evaluate objects that follow the community direction are "good" hubs of the community.

Another blog ranking algorithm named iRank was created by Adar, Zhang, Adamic, and Lukose (2004). This algorithm is the updated version of PageRank. As the first attempt to implement PageRank in the context of blogs, this algorithm allows the most cited blogs to become the hub of the blog environment and get higher popularity scores. Furthermore, pages containing popular postings would also be ranked high.

A more recent blog ranking implementation is the B2Rank algorithm (Tayebi, Hashemi, & Mohades, 2007). B2Rank works like the PageRank algorithm, but it assigns two scores to each blog: a personality score and an operation score. The most interesting part of this algorithm is that it uses two different types of links within the blogosphere, the blogroll and the permalink. If blog A has blog B on the blogroll list, it means that blog A believes blog B is an interesting blog and votes for it. A permalink is a citation for blog postings; it means that the author is only interested in specific postings, but perhaps not the whole blog.

Another contribution of B2Rank is that it defines the weight of links between blogs. For a blogroll network, the weight of each link is defined by the probability of moving from one page to another.

The weight of each link corresponds to two factors of linked blogs, including updating rate and average number of comments. They are both necessary to calculate the personality score. For a permalink network, the weights are computed according to the number of comments on linked posts and the delay time of the citation. The hypothesis is that a blog posting that gets more comments and earlier citation has a higher probability to be selected by a surfer. The operation score of a specific blog is the average of all its posting scores. So, the final rank score of each blog is:

$$B2Rank(x) = PE(x) \cdot EAR(x) \qquad (2\text{-}18)$$

where PE(x) is the personality score and the EAR(x) is the operation score. PE(x) calculation is very similar to the PageRank algorithm:

$$PE(x) = d(\sum(PE(y) \cdot NB(y,x))) + (1-d) \qquad (2\text{-}19)$$

This is also a recursive definition, just like PageRank. That is, the personality score of blog x is defined by all the blogs that have links to it. *NB(y, x)* is the probability that a reader moves from blog x to blog y. Similarly, the operation score calculates the posting relationships; the only difference is that this score should be normalized by the number of postings within a certain blog:

$$EB(e) = d(\sum(EB(b) \cdot NB(b,e))) + (1-d)$$
$$EAR(x) = \frac{\sum EB(e)}{num\_of\_posting(x)} \qquad (2\text{-}20)$$

Similar to linkage data, statistical user behavior data (such as clickthrough, dwell time and scroll time data) have been used and prove to be an effective indicator to compute a popularity score for ranking. The relationship between the explicit ratings of user satisfaction and the implicit measures of user interest was studied by Fox, et al. (2005). Two different Bayesian models were built to correlate different kinds of implicit measures and explicit relevance judgments for individual page visits and entire search sessions. The experiment shows that implicit behavior measures are strongly associated with users' satisfaction. Joachims (2002) employed clickthrough data to learn ranking function by using SVM, and his work proved that clickthrough data is a significant predictor of user interest. This work is based on the assumption that, in most cases,

user click behavior is not random, but based on a clear informed choice. Meanwhile, clickthrough data is available in abundance and can be recorded at very low cost. Similarly, Agichtein et al., (2006) incorporated noisy user behavior data into the search process, and the user data was used to train the ranking functions. In this approach, user interest is partially represented by statistical user behavior data.

Most existing popularity research and algorithms are based on PageRank, HITS algorithms and user behavior data. However, that causes some limitations:

- Too much focusing on timeline and the link structure of web pages may result in ignoring the important information within the content of queries and documents. This problem is especially important when a user cannot formulate a high quality query to represent his or her information need.

- Blog linkages (hyperlinks in web pages or relationships between blogs and postings) are different than in the traditional web medium. For instance, a blogger creates a link (blogroll or permalink or comment) probably for two different reasons: 1) the blogger is interested in the content of the target blog or posting, or 2) the blogger has a close relationship (e.g., friends) with the author (Furukawa, Matsuzawa, Matsuo, Uchiyama, & Takeda, 2006). Currently, little research clearly separates these two kinds of links.

- Mishne and de Rijke (2006) find that some web retrieval users (such as bloggers) are interested in context queries and news-related named entities, but they may be only interested in a specific aspect of this named entity. For example, with the name entity "Barack Obama," users may be interested in "his Chicago house" or "his book, The Audacity of Hope" for each time period. Some users cannot formulate a high quality query to express his or her interested topics, and popularity based ranking algorithms cannot catch these subtleties in users' interests.

Overall, we improve existing ranking algorithms to better satisfy user needs, as well as identify change in user interest over time.

2.4.2 Combining relevance and popularity based ranking

Given the relevance and popularity based ranking algorithms, we need to answer two different questions:

1. *How do we choose between relevance and popularity ranking for different queries?*
2. *How do we combine these two approaches to create a better ranking function?*

In recent years, leveraging the machine learning framework became the most popular way to combine different ranking algorithms. This method is known as 'learning to rank.' Several candidate ranking algorithms were aggregated using machine learning to build more effective ranking models (Bartell, Cottrell, & Belew, 1995; Burges, Ragno, & Le, 2007; Burges et al., 2005; Cao et al., 2006). A standard query level evaluation corpus, such as TREC, was utilized as the training data.

As the following diagram shows (Liu, 2009), in the learning to rank framework, the training set consists of a set of n training queries with their associated documents represented by feature vectors $x^{(i)}$ with their relevance judgments. Then one or more learning algorithms are used to learn the new ranking model or train the unknown parameter(s). It was found that the new ranking model can be used to predict the rank results of the new queries for testing purpose.

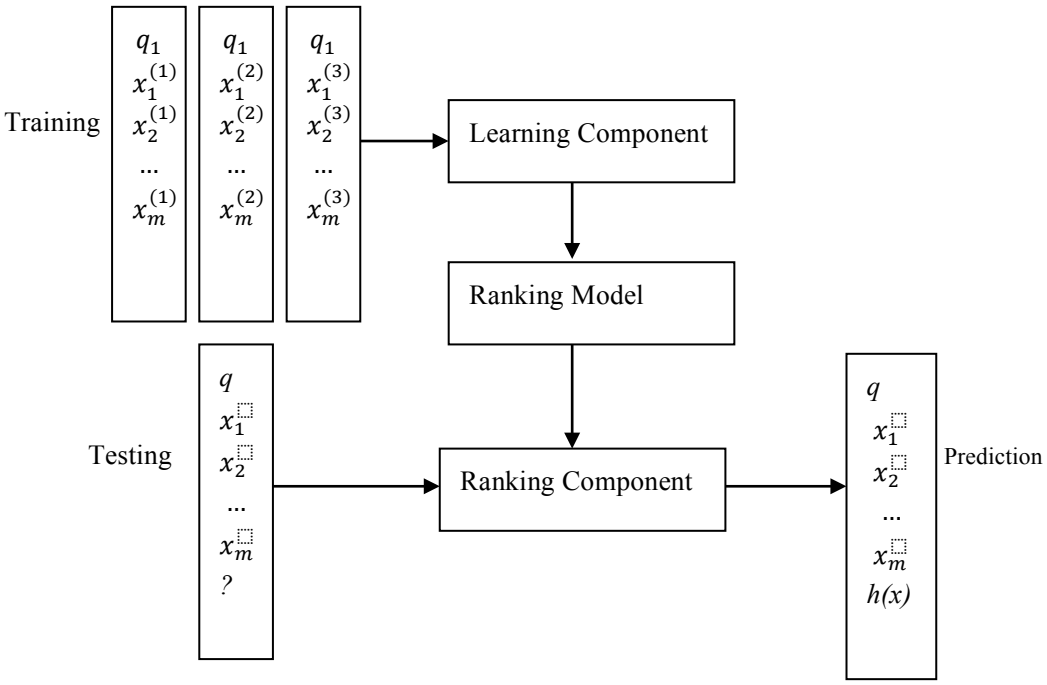

Figure 2-3. Learning to Rank Framework (T. Liu, 2009)

Trotman (2005) found two better ranking functions by aggregating well established ranking algorithms, including inner product (Witten, Moffat, & Bell, 1994), cosine (Harman, 1992), probability (Robertson & Spärck Jones, 1976) and BM25 (Robertson, Walker, Jones, Hancock-Beaulieu, & Gatford, 1995).

Learning to rank is an effective empirical methodology for integrating a list of different ranking algorithms from a machine learning perspective. Nevertheless, from a theoretical perspective, combining relevance and popularity ranking methods approaches is still a challenging task.

2.5 Topic modeling

Topic, defined as group or probability distribution of words or phrases, is frequently used in information retrieval and text mining research. In this thesis, the community interest model is built on a query and time dependent probability distribution over a list of topics. The topics on the "community interest topical space" can be very dynamic, due to the nature of users.

To address this dynamic characteristic, a topic extraction algorithm is used in this thesis, and the related topic extraction methods will be reviewed in this section.

2.5.1 Word clustering and topic

The earliest topic extraction methods extracted statistical topics by means of clustering algorithms. For instance, Lewis (1992) experimented with the performance of grouping those unambiguous, but semantically related indexing terms into clusters for classification tasks. An earlier similar experiment by Sparck Jones (1973) found that small clusters of low frequency terms were most effective, regardless of the clustering method used.

Topic models are a popular approach for representing the content of documents and collections from a semantic perspective. This approach stems from cluster hypothesis (Jardine & Rijsbergen, 1971), and a document is assumed to draw its words and phrases from one or more topics. Each topic is generally defined as the probability distribution, $P(t|z)$, over all of the words or phrases.

$$P(t|d) = \sum_{z} P(t|z) \cdot P(z|d) \qquad (2-21)$$

The popular word cluster methods are reviewed as follows:

Mutual information and entropy are frequently used to compute the relatedness between words by using their textual context. In these methods, the similarity of words is based on the co-occurrence data in the context, for example verb and noun pairs. (Brown, Pietra, deSouza, Lai, & Mercer, 1992) propose use of an algorithm based on Maximum Likelihood Estimation (MLE), that performs a merge that would result in the least reduction in average mutual information, while (Li, 2002) employed the Minimum Description Length (MDL) principle. These methods use a probability model that computes the probable existence of the co-occurrences among words or classes of words, for example, verb (v) occurrence with the noun (n):

$$P(n,v) = P(C_n, C_v) \cdot P(n|C_n) \cdot P(v|C_v) \qquad (2\text{-}22)$$

For any two words in the context, we can build the context vectors and define a distance metric based on the distance of the vectors in terms of (minimal loss of) Average Mutual Information (AMI), where AMI is the value of averaging the mutual information of individual word pairs. The idea at the basis of their clustering method is to find groups in which the loss of AMI is small. In general, the loss is smaller when the members of the group have similar vectors.

Alternatively, a taxonomy-based approach can be used for word clustering. Semantic similarity between words will be in respect to hierarchically structured lexical resources, namely IS-A or hypernymy / hyponymy relations, such as are used in WordNet (Fellbaum, 1998). Semantic similarity is evaluated in terms of the distance between the words (nodes on the tree structure) in the taxonomy: the shorter the path from one node to another, the more similar they are. Given multiple paths, the shortest path is understood as involving a stronger similarity.

In a real WordNet taxonomy environment, it has been noted that the “distance” covered by individual taxonomic links is variable, due to the fact that certain sub-taxonomies are much denser than others. To overcome this problem, the system normalizes the synset by using the maximum depth (from the lowest node to the top) in the taxonomy in which both words co-occur (Warin, 2004).

The easiest implementation is the Leacock-Chodorow method (Leacock and Chodorow, 1998). The similarity between the two concepts *a* and *b* equals the number of nodes along the shortest path between them, divided by double the maximum depth (from the lowest node to the top) in the taxonomy in which *a* and *b* occur.

$$Sim_{LCH}(a,b) = \max[-\log\frac{length(a,b)}{2D}] \tag{2-23}$$

Similar methods including Wu-Palmer method (Wu and Palmer, 1994) and Lin algorithm (Lin, 1998), which compute the similarity between words and phrases by leveraging lowest common subsumer (LCS, the lowest node subsuming / dominating them both concepts in the hierarchy):

$$Sim_{Wup}(a,b) = \max[\frac{2\times depth(LCS(a,b))}{length(a,b)+2\times depth(LCS(a,b))}] \tag{2-24}$$

$$Sim_{Lin}(a,b) = \max[\frac{2\times \log p(LCS(a,b))}{\log p(a)+\log p(b)}]$$

Natural Language Processing (NLP) is also used for word clustering and topic extraction, especially for defining the syntactic relatedness between words within the sentences. For instance, Jacquemin (1999) studied paradigmatic and syntagmatic relatedness between words. For paradigmatic, two words can be similar if they can substitute for each other in a particular context without violating any semantic rules of the sentence. For syntagmatic, the similarity is based on typically co-occurring terms within the same context. By using this method, nouns, verbs and noun + verb pairs could be clustered to create semantic topics.

Word and document clustering is a popular technique in information retrieval research. For instance, in the earlier stage, Voorhees (1985) used clustering for basic retrieval experiments to test clustering hypothesis for information retrieval task, but the results are inconsistent. Liu and Croft (2004), Xu & Croft (1999) used clustering in a different way, which integrated cluster based topics with language modeling and smoothing, which achieved positive results. Evans, Huettner, Tong, Jansen, and Bennett (1999) used clustering for interactive relevance feedback. Existing studies show that word or document based clustering can be an effective method to improve retrieval performance.

2.5.2 Latent Semantic Indexing

Traditional information retrieval or text mining systems use bag-of-words or phrases as the features to represent each document in the collection, and the multinomial word distribution has long been used to describe the content of the query, document or collection. However, word level document indexing is more like a type of statistical representation, instead of semantic representation. Meanwhile, the fundamental deficiency of systems' ability to deal with synonymy and polysemy effects (Deerwester, et al., 1990) may threaten the performance of retrieval system, because bag-of-words indexing assumes that each word is independent from the others.

Latent semantic indexing (LSI) was first introduced by Deerwester, et al. (1990), and can effectively find the relatedness between words by using the observed occurrence information in the corpus matrix. LSI uses the high dimensional word frequency document vector space representation matrix as input and then employs the linear dimension reduction, Singular Value Decomposition (SVD), to project the high dimensional matrix into a lower dimensional one. The new reduced lower dimensional matrix is called latent semantic space, namely, each component in the new matrix can be used as a latent semantic topic for retrieval usage.

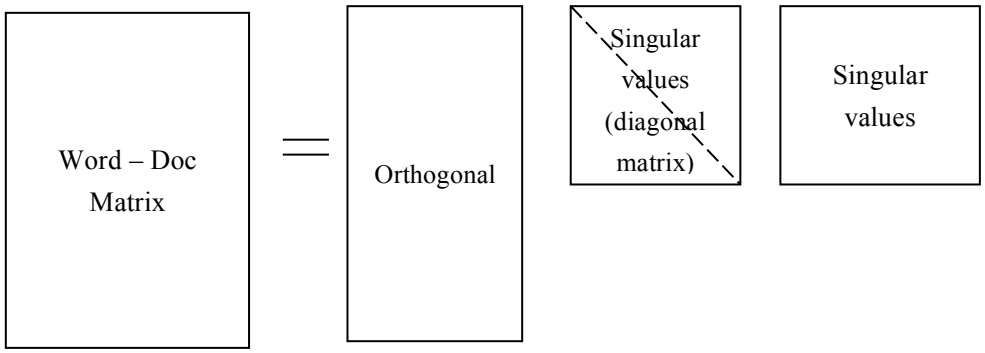

Figure 2-4 LSI Model

For LSI, the rectangular word-document matrix is decomposed into three matrices as the above diagram and formula shows. T_0 and D_0' have orthonormal columns and S_0 is a diagonal matrix.

If only the largest k singular values (in S_0) are considered, the model X can be reduced into $\widehat{X}$, which reflects the most important information and associative patterns in the corpus (e.g. the relatedness between words) while filtering the noisy information in the data. The LSI model can tell the relationships between words, documents and word document pairs. For instance, the dot product between two rows of the matrix $\widehat{X}$ reflects two terms having a similar context of occurrence across the document; similarly, the dot product of two columns in $\widehat{X}$ reflects the relationship between two documents.

LSI also performs noise reduction and has the potential benefit to detect synonymy and polysemy as well as words that refer to the same topic by using the context information in the corpus matrix. It has been applied by different fields related to information retrieval, such as Foltz (1990) for information filtering, Praks, Dvorsky, and Snášel (2003) for image retrieval, and Dumais, Letsche, Littman, and Landauer (1997) for cross language retrieval.

In many applications, LSI has proven to result in more robust word and topic processing than bag-of-words, especially when there is little overlap between queries and documents. However, it also has a number of deficits, for instance, when integrating LSI to traditional vector space model, it is hard to express negations and hard to use Boolean conditions. Meanwhile, employing the latent relatedness between words in retrieval may result in higher recall but lower precision.

In contrast to standard Latent Semantic Indexing, the probabilistic variant has a more solid statistical foundation based on latent random variables. Hofmann (1999) defines a generative model based on LSI called Probabilistic Latent Semantic Indexing (PLSI). Retrieval and text mining experiments on a number of test collections and tasks indicate substantial performance improvements over classical LSI.

A generative model for documents is based on the assumption that each document in the collection is a mixture of a list of topic probability distributions and the simple probabilistic sampling rules that describe how words in documents and collections might be generated on the basis of latent random variables (Steyvers & Griffiths, 2007). The most important steps for PLSI are to model the topic-word distributions $\emptyset$ and the topic distributions θ given a document. In the progress of training a generative model, the goal is to find the best set of latent variables that can

best explain the observed data, for instance, observed words in document or collection, assuming that the model actually generated the data. The extracted model, such like topic probability distribution, can be used to infer the unobserved data.

PLSI is a proper generative model, as the following diagram shows, between document and words, if we assume there is another latent variable as topic z, with the conditional independence assumption on the latent class z,

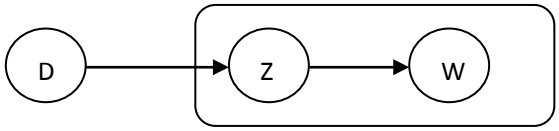

Figure 2-5. Probabilistic Latent Semantic Indexing (PLSI)

the document generation probability of a word given a document can be written as (2-21), $P(w|d) = \sum_z P(w|z) \cdot P(z|d)$ In this formula, $\mathrm{P(w|z)}$ is the word probability given a topic and $\mathrm{P(z|d)}$ is the topic probability given a document. Given the likelihood principle, for all the documents in the corpus, the maximization of the log likelihood function can be written as

$$L = \sum_D \sum_w n(d,w) \cdot logP(d,w) \qquad (2-25)$$

While, given latent topic variable, $\mathrm{P(d,w)}$ can be:

$$P(d,w) = \sum_z P(z) \cdot P(w|z) \cdot P(z|d) \qquad (2-26)$$

The Standard Expectation Maximization (EM) (Dempster, Laird, & Rubin, 1977) is used for the procedure for maximum likelihood estimation. The EM algorithm performs the estimation process iteratively, which can be used for unobserved data.

Overall, PLSI, based on the latent variable topic with a better statistical process, outperforms the LSI model (Hofmann, 1999). For IR studies, Azzopardi, Girolami, and Van Rijsbergen (2004) used PLSI to construct thesauri through automatic synonym acquisition.

The extracted topics can be used to describe the contents of a document or collection: the high probability topics and words within the topics can be viewed as a loose description of the document and collection, and the more sophisticated topic models can provide better descriptions

(Yi & Allan, 2009). In the meantime, some researchers have claimed that topic models can improve information retrieval by matching queries to documents at a semantic level, such as (Steyvers & Griffiths, 2007).

2.5.3 Latent Dirichlet Allocation

PLSI is the first statistical generative topic modeling algorithm based on the hypothesis that a document is a mixture of latent variables. However, the PLSI model does not make any assumptions about how the mixture weights θ, e.g. topic distribution over documents, are generated, making it difficult to test the generalizability of the model to new documents and making the parameters training process difficult (Steyvers & Griffiths, 2007).

Blei, Ng, and Jordan (2003) proposed Latent Dirichlet Allocation (LDA), which is a generative probabilistic model in the hierarchical Bayesian framework. LDA extends PLSI by introducing the Dirichlet prior on θ, the new generative model called LDA. As a conjugate prior for the multinomial topic distribution, the Dirichlet distribution assumption has some advantages, which can simplify the problem. The probability density of a T dimensional Dirichlet distribution over the multinomial distribution $p = (p_1, p_2 \ldots, p_T)$, where $\sum \alpha_j = 1$, is defined by:

$$Dir(\alpha_1, \alpha_2 \ldots \alpha_T) = \frac{\Gamma(\sum_j \alpha_j)}{\prod_j \Gamma(\alpha_j)} \prod_{j=1}^{T} p_j^{\alpha_j - 1} \qquad (2-27)$$

where $\alpha_1, \alpha_2 \ldots, \alpha_T$ are the parameters of Dirichlet distribution, which can be simplified by a single value α_{LDA}. The value of α_{LDA} is dependent on the number of topics K. Each hyperparameter α_j can be the prior observation for the number of times topic j is sampled in a specific document before having observed any actual words from that document. Similarly, the Dirichlet distribution on β_{LDA} prior on $\emptyset$, e.g. the word distribution over topics, the hyperparameter β_{LDA} can be interpreted as the prior observation count on the number of times words are sampled from a specific topic before any word from the collection is observed. The LDA topic modeling process is shown as the following diagram. The common settings of symmetric Dirichlet priors in the LDA estimation with $\alpha_{LDA} = 50 / k$ and $\beta_{LDA} = 0.01$ will be

used for this thesis, and according to Wei and Croft (2006), the retrieval performance is not very sensitive to the values of these parameters.

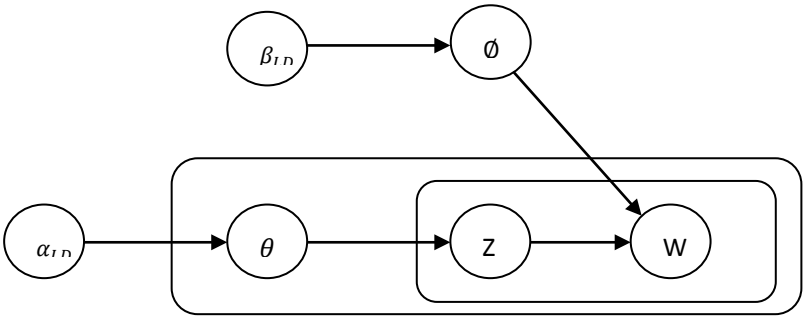

Figure 2-6. LDA

LDA is the most popular topic extraction algorithm in recent years and it has been successfully used in information retrieval and text mining research to effectively characterize the content of a document or a collection. For instance, Zhou, Bian, Zheng, Zha, and Giles (2008) proposed a generative model for social annotations based on LDA topic models. Sivic, Russell, Efros, Zisserman, and Freeman (2005) used LDA for image retrieval. Titov and McDonald (2008) implemented the new model for opinion analysis based on LDA.

LDA is also frequently used in current information retrieval related research. For example, Azzopardi, et al. (2004), Wei and Croft (2006) improved on the classical language model from topic modeling smoothing by integrating LDA topic component.

2.6 Evaluation methodology

In this section, we will briefly introduce the popular evaluation measures in traditional and web based information retrieval contexts.

Most experimental information retrieval studies focus on developing innovative methods and algorithms to better match and rank the retrieved results, and it is critically important to utilize scientific evaluation criteria to judge if one method is significantly better than another. Among popular evaluation methodologies, there are several basic rules (Buckley & Voorhees, 2000):

- The test collection should have a reasonable number of testing queries. Sparck Jones and van Rijsbergen (1976) suggest a minimum of seventy-five queries, while the TREC

program committee has used twenty-five queries as a minimum and fifty queries as the norm (Voorhees & Harman, 2000).

- The experiment should use a reasonable evaluation measure. The common evaluation metrics include average precision, R-precision, precision-at-(top) n documents, and normalized discounted cumulative gain (NDCG).

- The conclusions of whether retrieval performance improves or not should be based on a reasonable notion of difference. Sparck Jones (1974) suggested that a difference in the scores between two different algorithms should be greater than 5% to be noticeable.

Precision and recall have been used as the key metrics to evaluate information retrieval systems and algorithms for a long time. Most existing precision and recall based ranking evaluation is based on binary relevance judgments, namely every, or a subset of, retrievable documents are recognizably "relevant" or "not relevant." The precision and recall rate in information retrieval is defined by the following formulas:

$$Precision = \frac{|Relevant\ Documents \cap Retrieved\ Documents|}{|Retrieved\ Documents|}$$

$$Recall = \frac{|Relevant\ Documents \cap Retrieved\ Documents|}{|Relevant\ \ Documents|} \qquad (2-28)$$

Intuitively, recall in IR is the measure of the completeness of retrieval, while precision describes the purity and effectiveness of retrieval. Even though achieving high precision and recall simultaneously is preferable, empirical studies of information retrieval show that a tradeoff between precision and recall is unavoidable with a tendency for precision to decline as recall increases (Buckland & Gey, 1999). The relationship between precision and recall has been studied since the 1970s (Bookstein, 1974; Buckland & Gey, 1999; Cleverdon, 1972; Gordon & Kochen, 1989; Heine, 1973; S. E. Robertson, 1975). F-measure is frequently used to quantify retrieval performance by combining precision and recall.

$$F_\beta = \frac{(1+\beta^2)Precision \cdot Recall}{\beta^2 \cdot Precision + Recall} \qquad (2-29)$$

When recall and precision are evenly weighted, $\beta = 1$, it is known as $F1$ measure.

In most IR evaluation methods, we need to evaluate more than one query, and mean average

precision (MAP) is one of the most stable metrics. MAP across different testing query set (size = Q) is defined as:

$$MAP = \frac{\sum_{q=1}^{Q} AveragePrecision(q)}{Q} \qquad (2-30)$$

Where average precision for each query q is defined as:

$$AveragePrecision(q) = \frac{\sum_{r=1}^{N}(P(r) \cdot rel(r))}{number\ of\ relevant\ docs} \qquad (2-31)$$

In formula 2-31 average precision is defined by each of the relevant documents in the ranked list. *P(r)* in the formula is the given cut-off rank and *rel(r)* is the binary relevance judgment for document at rank *r*.

Traditionally, the recall-precision plot is often employed to characterize performance of the information retrieval system and algorithm performance. For very large and dynamic document collections, such as the web search environment, it becomes impossible to get accurate recall estimates, since they require relevance judgments for a large document collection (Joachims, 2002). For this reason, we need another evaluation methodology to satisfy the requirement of search engines.

In the web environment, users tend to peruse the first page of retrieved pages, but rarely move to the second and almost never look at the third. Anh and Moffat (2002) proposed Precision-at-document-n, which focuses on the precision of the top n rank results in the result collection.

The Discounted Cumulative Gain (DCG) ranking evaluation (Järvelin & Kekäläinen, 2002) is another good indicator to evaluate ranking result. DCG is a measure that gives more weight to highly ranked documents and allows incorporation of different relevance levels (highly relevant, relevant, and not relevant) by giving them different gain values. The formula for this is:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i}-1}{log_2\ (1+i)} \qquad (2-30)$$

Choosing the right evaluation method and reasonable testing query collection size is important for evaluating the new algorithm. In Buckley and Voorhees's (2000) work, they evaluated different evaluation methods by calculating the error rate while verifying topic set size. The evaluation

methods they analyzed include precision-at-n, recall (1000), precision at 0.5 recall, R-precision and average precision. In their experiment, they found that:

1. Twenty-five topics are just barely enough for an experiment, but fifty topics are stable with the error rate less than 2-3%.
2. This suggests that 100 queries is a good target number for an experiment measuring Precision-at-20.

A similar experiment has been implemented by Sakai (2006).

However, most existing evaluation methodologies are designed for static evaluation purposes, which means user's relevant judgment doesn't change very often. For this thesis, a dynamic community interest model is extracted to satisfy a user's up-to-date information needs, but an updated (dynamic) evaluation method is also needed to judge the new ranking algorithms.

2.7 Conclusion

Since this thesis is attempting to improve ranking performance by leveraging dynamic community interest, it is helpful to understand the existing ranking methodologies. It is encouraging to know that statistical user or community information has been successfully employed to improve ranking performance.

The community interest in this thesis is defined as a degree or a probability distribution of a list of real-time query centric topics. The use of topic extraction is an important component of the experiments. Based on recent studies in information retrieval (Wei & Croft, 2006), LDA is an effective topic modeling algorithm, which can be integrated with well-established retrieval model, such as language model.

However, few of the earlier studies have explicitly used community interest in order to improve ranking. This thesis attempts to make use of dynamic computational community interest model extracted from blog data to better understand users' real-time preferences given a query and improve ranking performance.

Chapter 3: Methodology

3.1 Introduction

As stated in Chapter 1, the area of inquiry in this thesis is the innovative *community interest* based ranking method. Three research questions derived from the research goals have focused the inquiry:

RQ1: What is community interest? And can we extract and computationally model real time community interest from user textual data?

RQ2: In what ways can real-time community interest be used to rank the retrieved results?

RQ3: How can we evaluate real-time community interest ranking results? And can the community interest based ranking method improve results over existing ranking methods?

In this chapter, we will explain the research methodology in detail. In section 3.2, how to utilize user-oriented text corpus to represent real-time user interest and how to effectively extract the statistical community interest topic model within the user-oriented text data is discussed (RQ1). In section 3.3, the community interest model (extracted from RQ1) used as an indicator to rank the retrieved results will be discussed. Two ranking methods will be introduced: *Community Interest Vector (CIV)* and *Community Interest Language Model (CILM)*. And last, an innovative evaluation method will be explored in section 3.4 to evaluate the dynamic ranking results, and a preliminary experiment with evaluation results will be analyzed.

In section 3.2, we will mainly address the problem of what is community interest and the method to model community interest. However, it is difficult to directly evaluate if the extracted community interest model is an accurately representation of community interest. So, we will combine this question with RQ3 and hypothesize that, if the community interest model is accurate, the ranking result based on the likelihood of interest will be positively evaluated by user and the interest based ranking should be better than the baseline ranking methods.

As mentioned in the first chapter, currently there are two different approaches for information retrieval ranking methodologies: *relevance based ranking* and *popularity based ranking*. In this

thesis, another ranking method based on user (community) dynamic interest over the content of the documents is proposed. In Table 3-1, these three ranking approaches are compared.

Ranking Method	**Query**	**Community or User**	**Time**
Popularity	Independent	Dependent	Semi- Dependent
	P (popular | doc) e.g. PageRank, ClickThrough, behavior based ranking		
Relevance	Dependent	Independent	Independent
	P (relevance | doc, query) e.g. VSM, language model, BM25		
Interest	***Dependent***	***Dependent***	***Dependent***
	P (interest | doc, query, time, community)		

Table 3-1. Comparison of three different ranking methodologies

The interest based ranking method is dependent on a query, the community/user, and time. Extracting and modeling community interest is a challenging task. Since users create millions of pieces of textual information every hour, such data is available in abundance. In the rest of this chapter, the method of building a community interest model for ranking by utilizing user-oriented real-time text data will be explained.

3.2 Community Interest Generation

3.2.1 Definition of community interest

In the existing information retrieval research, user need has been studied by the different methods outlined in Table 3-2.

Obviously, a user's query provides the most direct evidence of a user's information need. However, as mentioned in Chapter One, the gap between a keyword query and an information need, represented by α, is an intangible factor which is difficult to measure. Because of the challenge of factor α, most existing retrieval and ranking algorithms rely solely on the query to

model a user's information need. However, since a query is often extremely short, the user model constructed based on a keyword query is inevitably impoverished.

Method	*Describe*	*Example*
Feedback	Explicitly or implicitly add additional terms to the query based on previously identified relevant or non-relevant results	Salton and Buckley, (1990)
Query-log analysis	User generated query log is the representation of the information need for query recommendation or clustering purposes	Baeza-Yates, et al., (2004b)
Collaborative filtering	Recommend the information by discovering the similar users in certain community	Breese, Heckerman, and Kadie (1998)
Personalization	Personalize the search or rank result based on an existing user profile	Qiu and Cho (2001)

Table 3-2. Existing research for user requirements in IR

As background for this study, user or community interest can be viewed as a kind of dynamic information need or context for an information retrieval task and each community interest model can be used as the user-oriented search context for the target query. Users input the same query (especially for short queries) at different times (different months, days or hours), often for different reasons, and the corresponding relevance judgments, interest judgments and ideal ranking result might change over time. For instance, if a user is interested in news about the playoff chances of her favorite sports team, news of today's win will be of interest today but may well not be of interest next week and the ideal rank list of documents may change over time even though neither the document nor the information need has changed. For these kinds of information needs, dynamic or time variant ranking functions would be desirable.

Another example, "*Obama*" was a popular query in 2008 and 2009 query logs, but user or United

States community interest toward this query may change, as Table 3-3 shows.

Query	*Time*	*Community's Interest (topics)*
"Obama"	August, 2008	*Presidential candidate competition within Democrats…*
	October, 2008	*Presidential candidate competition between Democrats and Republicans…*
	December, 2008	*Plan for economic crisis and bailout…*

Table 3-3. Community interests change over time for query "Obama"

If we are to effectively extract community interest toward each query, for example by extracting real-time interest models at a regular time interval, i.e. every thirty minutes, especially for short and ambiguous queries, the gap between the query string and users' information needs will be lessened, as the following diagram depicts:

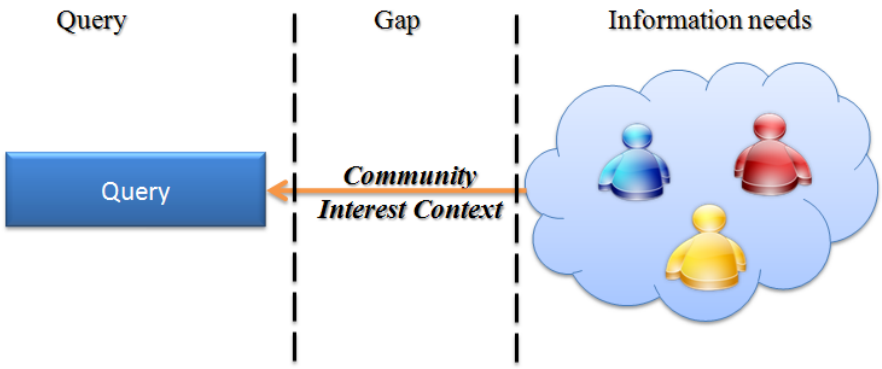

Figure 3-1. Community interest context for query

In order to characterize community interest, it is necessary to introduce query specific *community interest topic space*, which is defined as follows:

> ***Community interest topic space (query specific)****: a list of current topics that a community is likely interested in for a given query.*

"Community interest topic space" is a sub-collection of "fact topic space" for a given query. If a query is the "protagonist" of a given set of news, we could find the corresponding topic(s) of the news in the "fact topic space", but the community may or may not be interested in a given topic (event), and this specific topic may or may not appear in the "community interest topic space". The protagonist is the main actor in a posting. In this thesis, a protagonist is not necessarily a

person; instead, it is a query, which could be a person, a location, an event or something else.

For a given time and a given query, the community may be interested in more than one topic (with different degrees of interest), and the community's interest can be defined as follows:

> ***Community interest***: *a distribution of the degree or probability of interest in the community interest topic space for a given query at a given time; each point in this distribution mirrors the current real-world community's interest toward a specific topic relative to the target query.*

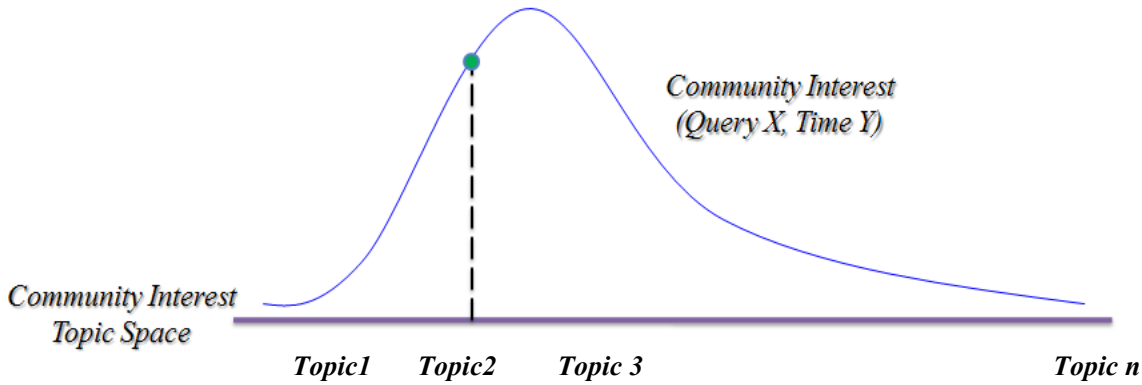

Figure 3-2 Community interest (distribution) for query X, time Y

As Figure 3-2 shows, the community interest for a specific query X at time Y is defined as a distribution over the community interest topic space, and the point in this diagram demonstrates the probability (or degree) that a community takes interest in *Topic 2*. Based on this definition, there are two essential steps in order to extract community interest. First, the community interest topic space should be identified; and secondly, the probability or degree of each topic in the community interest topic space should be computed to estimate real time interest distribution. Both the community interest topic space and community interest distribution can be dynamic and change frequently.

3.2.2 User generated textual data and community interest

Based on the definition of community interest, there are five fundamental components to modeling

the interest: *query, community, time, topic, and weight or probability (of the topic)*. In existing IR research, most user or community ranking models are trained by statistical user behavior data, such as clickthrough, hyperlink network or dwell time. The dynamic community interest model and interest distribution for information retrieval ranking is trained using real-time user-generated textual data.

In the Web 2.0 context, users may generate different kinds of chronological text data, such as blogs, selected news, or comments to express their opinions. A hypothesis of this study is that a large amount of user-oriented chronological textual data can represent the overall opinion and interest of the community of a target query. A simple example is the 2008 presidential election. As the following diagram shows, the number of blog postings containing "Obama" and "McCain" changed over time (data from Yahoo! Buzz, http://buzz.yahoo.com, from 2008-10-11, before the election, to 2008-11-10, after the election). In this case, "Obama" and "McCain" were the protagonists of the dynamic blog posting collections, and these collections reflect the interest change over the query "Obama" and "McCain".

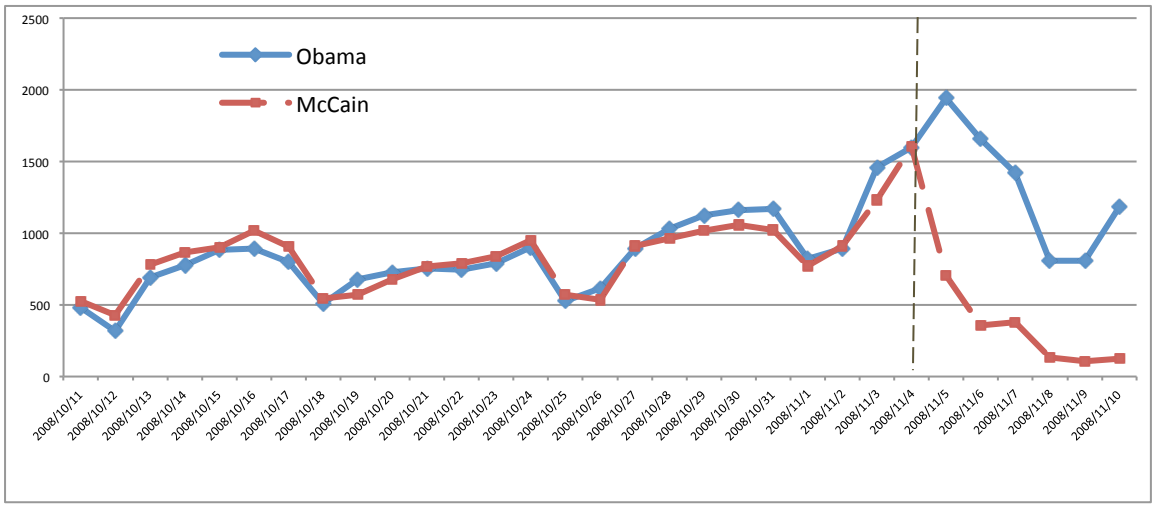

Figure 3-3. 2008 US president candidates related blog postings

It is shown in Figure 3-3 that, before election day (11/4/2008), the numbers of postings about the two candidates were almost equal, but after the election, because of the result, the gap between the winner and the loser significantly increased, and represented real world interest. Similar blog research about the 2004 presidential election can be found in Adamic and Glance (2005).

Compared with statistical user data, such as query log and search session data, user generated chronological textual data has the following advantages:

1. *User generated chronological textual data contains sufficient semantic content for topic extraction as well as the generation of community interest topic space.*

2. *User generated chronological textual data is time dependent, and it can be classified into "current collection" and "historical collection," representing "current community interest" and "historical community interest." For some other time oriented dataset, like query log, we can hardly extract the subtle interest change from semantic perspective.*

3. *Compared with statistical user behavior data (such as hyperlink and clickthrough), user generated textual data may be less noisy, resulting in higher representayion fidelity.*

But user generated chronological textual data also have some disadvantages:

1. *Compared with statistical data, textual data is difficult to analyze, and extracting semantics (topics) from text data is not an easy task.*

2. *User generated chronological textual data is used to train the community interest model which is then used to rank the documents. However, some user oriented text data (e.g. blog postings) are written in informal/colloquial language. The gap between colloquial and formal language may negatively affect the performance of ranking because of some low quality features in the community interest topic space. For instance, taking news ranking as a case, user currently interested in topic X, but the word distribution of topic X in news corpus and blog corpus may or may not be same.*

For a specific piece of user oriented textual data, such as a blog posting *P*, if query *X* is the protagonist of this posting at time *Y*, we can assume that *P* is a sample of community interest for query *X* at time *Y*, and *P* can be used as an unit to train the community interest model.

In the ideal world, determining if a blog posting is mainly focused on the target query may increase the accuracy of the interest model by filtering out the noisy data. For instance, the word "Obama" shown in one posting does not necessarily mean that "Obama" is the protagonist of the

posting. However, removing the noisy data will also significantly reduce the size of the training data. Some relevant posting data could be inevitably filtered out. In preliminary experiments, we found that training data size is very important for community interest modeling (detailed in section 3.4). As a result, this thesis makes the assumption that if the target query appears in a posting, this posting may have a strong semantic relationship with the query, and this posting can be used to train the community interest model.

3.3 Community Interest Ranking

3.3.1 Community interest topic space extraction

As defined in section 3.2.1, community interest topic space is a list of query-specific topics that a community is highly likely interested in at the current moment. This section will explain the method of extracting topics from user oriented text by using topic extraction algorithm.

In classical information retrieval and text mining systems, the bag-of-word assumption is frequently used as the basic feature to index text data, and both query and document are represented as empirical unigram distributions over the vocabulary. Recently, more and more researchers have begun to add topic modeling in order to represent the content of the document and collection in information retrieval studies. In this study, the topic is defined as a probability distribution over the vocabulary, and the topic model can be used to characterize the content of one or multiple documents or collections (Yi & Allan, 2009).

Based on the definition of community interest space, the most current topics can be extracted from recent (e.g. the past few hours or today) user generated text documents. In this thesis, users' blog postings are used as the training data, and are separately indexed from (retrieval system) document collection. Because community interest topic space and the community interest model are query dependent, the most current blog postings containing the target query string are used as the training corpus, and the query in each posting in this corpus can be viewed as the protagonist (as mentioned in last section). *CQC (Current Query-centric Collection)* is used to represent this (current) training blog posting collection for each target query.

If we assume that the postings in the *CQC* incorporate a fixed number of latent topics, we can

proceed to extract these topics for community interest space. There are various techniques to perform this topic modeling step, and we chose an off-the-shelf public domain algorithm Latent Dirichlet Allocation (LDA) (Blei, Ng, & Jordan, 2003). In a nutshell, LDA is similar to probabilistic Latent Semantic Analysis (Thomas Hofmann, 1999) in that it decomposes the posting-by-features matrix into a *document-by-topics, matrix-α*, and a *topics-by-features, matrix-β*, illustrated in Figure 3-4. Azzopardi, Girolami, and Rijsbergen (2004), Wei and Croft (2006) have shown that using the LDA model for information retrieval tasks is feasible with suitable parameter settings, and LDA topic modeling performance consistently outperforms the traditional cluster based approach, illustrated by Liu and Croft (2004).

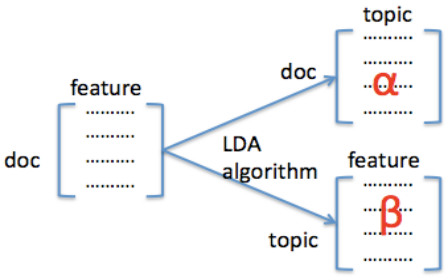

Figure 3-4. LDA topic modeling

LDA is a generative probabilistic model in the hierarchical Bayesian framework, and the topic proportions are randomly drawn from a Dirichlet distribution. As the above diagram shows, traditional document indexing systems represent each document as a vector of features, e.g. bag-of-words or entities. By using LDA, the document-feature matrix can produce two different matrices: matrix-α contains the document (posting) – topic probability distributions, i.e. each row represents the probability of the topic given the posting *P (topic | posting)*. Matrix- β contains the topic-feature probability distributions, i.e. each row represents the probability of each feature given the topic *P (feature | topic)*.

In the LDA model, the document corpus is generated by the following process:

1. *For z = 1: k, where k is the fixed number of latent topics, draw parameters for a multinomial distribution z for each topic z from a Dirichlet distribution with hyperparameters β. z models the relative frequencies of features in topic z.*

2. *For each document d, draw parameters for a multinomial distribution d from a Dirichlet distribution with hyperparameter α. d models the relative frequencies of topics in document d.*

3. *For each feature (e.g. word) w in document d,*

    a. *Draw one topic indicator* z_n *from the multinomial distribution d.*

    b. *Given* z_n*, draw a feature (word) w from the multinomial distribution* z_n.

The following table is an example of LDA topic model (topics-by-features probability distribution, matrix-β) from a user blog posting collection:

Swimming Topic		***Russia War Topic***		***Gymnast Topic***	
Michael_Phelps	0.024279	Georgia	0.008965	Liukin	0.011639
Phelps	0.017913	Russia	0.007939	Nastia_Liukin	0.011465
Swimming	0.011785	spanish	0.00598	Gymnast	0.010945
Gold_medal	0.011387	russia	0.005513	Shawn_Johnson	0.009469
Swimming	0.010114	war	0.005047	Gymnastics	0.007647
200m	0.009398	Georgia_U.S._state	0.003611	Johnson	0.00565
Swim	0.009398	states	0.003368	Age	0.005477
Record	0.008443	russian	0.002342	Nastia	0.004869
Phelps	0.008443	georgian	0.002342	Gymnast	0.004782
Meter	0.008045	bush	0.002248	Born	0.003828
Freestyle	0.007409	George_W._Bush	0.002062	Young	0.003567
World_record	0.006215	soviet	0.001968	http	0.003567
Relay	0.005021	ning	0.001968	Womens	0.003307
Water	0.004066	fight	0.001782	nastia	0.003307
rebecca_soni	0.003668	did	0.001782	Old	0.00322
100m	0.003509	Spain	0.001782	Father	0.00322
Swimmer	0.003509	iraq	0.001688	chinese_gymnasts	0.00322
Medley_swim	0.003509	oil	0.001595	Years	0.003133
Ryan_Lochte	0.003509	russias	0.001595	Uneven_bars	0.00295
Jones	0.003271	south	0.001502	Gymnastics	0.002786

Table 3-4. Three topic distribution for query the "Olympic" on 2008-08-11

The above table shows three sample topics extracted from the 2008-08-11 blog posting collection (1086 blog postings, number of topic N = 30, query = "*Olympic*"). An named entity recognition algorithm (Brzeski et al 2007) is used to identify phrases as feature. Each topic is represented by features (bag-of-words and entities), and the probability of the feature given topic *P (feature |*

topic). The top 20 features of each sample topic are printed in the table.

Based on the topic-feature probability distribution, the learned LDA model (from blog data) can be used to infer the topic distribution in a new document, and this is important for ranking, by, for example, projecting a candidate retrieved document into the *community interest topic space* to represent the probability that the community is interested in this document. Given a new unseen document, by inverting the LDA generative process, we can obtain the topic probability distribution of the new document. Each dimension represents the relative degree (or probability) for each topic that the community is interested in given a document:

$$TV(doc_x) = \{P(topic_x|doc_x), P(topic_x|doc_x) \dots \dots P(topic_x|doc_x)\} \qquad (3\text{-}1)$$

$TV(doc_x)$ is the topic vector of the given document X, while the $P(topic_m|doc_x)$ score represents the probability that $topic_m$ is a correct descriptor of the given document.

3.3.2 Community interest topic space parameter setting

Using a real-time blog posting collection as the community interest training corpus has some advantages as mentioned in the last section, however, finding the optimized parameter settings is challenging and important in this research. In most existing information retrieval research, automatic evaluation is used to train the parameters. But in this thesis, the dynamic community interest model is used to rank real-time retrieval, and evaluation is expensive to apply.

The first question for this research is *how often does a retrieval system need to update the community interest model for ranking*? As the LDA based topic modeling is employed by this study, complexity is a big concern because of its high computational costs. Meanwhile, because community interest models are query dependent, the number of community interest training corpora correspond to the number of the queries. Queries have differing numbers of blog postings in a certain period of time, which results in different training corpus sizes. So, another related question is *how many training postings should each query use*?

If query *X* is very popular in the community blogospace, it may have a large number of postings for a short period of time, and this query community interest model should be updated after a relatively short period of time. For those queries in the experiment, the maximum number

(num_{max}) of postings in the training corpus is fixed, and when the number of daily (past 24 hours) postings of a query is larger than num_{max}, the training size is fixed to num_{max}, which corresponds to the past *t* hours and where *t* < 24.

Another parameter used is num_{min}, which describes the minimum number of required postings for the training corpus. There are two reasons to employ num_{min}. First, most topic modeling algorithms, such as LDA, need a reasonable number of documents to train the topic model. Second, too few postings do not satisfy the required sampling of community interest for the target query.

In sum:

$$if\ num_{24hours} > num_{max}:\ num_{train} = num_{max};\ t_{train} = t(num_{max})$$

$$if\ num_{max} > num_{24hours} > num_{min}:\ num_{train} = num_{24hours};\ t_{train} = 24$$

$$if\ num_{24hours} < num_{min}:\ Query\ will\ be\ ignored \qquad (3\text{-}2)$$

If there is only a small number of blog postings ($num_{24hours} < num_{min}$) for the target query in the past 24 hours, this query is ignored by the current interest based ranking algorithm, because it is important to sample enough data to generate an accurate community interest model, and we are targeting on the subtle user interest change over a short amount of time (i.e. 24 hours). In actuality, it is possible to deal with this problem by using a "pseudo query" as the protagonist to train the target interest model. For instance, a query clustering algorithm was studied by Baeza-Yates, Hurtado, and Mendoza (2004a), Beeferman and Berger (2000), Liu, Qin, Chen, and Park (2008), Wen, Nie, and Zhang (2002), and it was found that the query cluster can be used as a "pseudo query." In this way, the interest model corresponds to the "pseudo query" instead of a single query, and the number of training postings of the query cluster is significantly increased. However, some popular query cluster algorithms, e.g. Wen, Nie, and Zhang (2002), are based on the user session and clickthrough data, which are hard to access. Despite some studies on pseudo queries, more research needs to be done, and this thesis only focuses on the query level interest model based ranking method.

In a preliminary experiment, nine popular queries were sampled from the top queries in the Yahoo

query log (2008 October and November query log from Yahoo web search). The blog training data was collected from Yahoo Buzz[2], a user-generated news-related blog service. Once again, because of real-time user evaluation purpose, we were not using a standard dataset, such as TREC blog and New York Times Data. When $num_{max} = 1{,}000$, and $num_{min} = 150$, the size of an average training corpus is as listed in the table below (data collected in the second week of November 2008).

In Table 3-5, two queries, "*economy*" and "*Obama,*" are very popular in the blogospace and the average corpus size (for 24 hours) is more than 1,000 (num_{max}) for the five experiment days. As a result, their interest training corpus size was limited to 1,000. Their average interest model coverage times correspond to 18.3 hours and 16.1 hours.

Query:	***Average Training Size***	***Average cover time***
Bush	517.7 postings	24 hours
Economy	***1000 postings***	***18.3 hours***
Obama	***1000 postings***	***16.1 hours***
McCain	245.3 postings	24 hours
Wall Street	405.3 postings	24 hours
Iraq	258.7 postings	24 hours
Google	284.3 postings	24 hours
Microsoft	177.6 postings	24 hours
Movie	440.7 postings	24 hours

Table 3-5. Average training corpus size & training average time

Another set of parameters is found in the LDA-related topic extraction part, which is closely related to community interest topic space generation. In LDA, the parameters α_{LDA}, β_{LDA} are the Dirichlet distribution priors for multinomial topic and document distributions. The common settings of symmetric Dirichlet priors in the LDA estimation with α_{LDA} = 50 / K and β_{LDA} = 0.01 is used in this research. According to Wei and Croft (2006), the retrieval performance is not very sensitive to the values of these parameters.

Compared with α_{LDA} and β_{LDA}, the number of topics, K, is important in order to extract an accurate community interest topic space. Based on the definition, community interest topic space is composed of a list of threads of interests (topics) with respect to the target query. One of the key

[2] *buzz.yahoo.com*

questions in this thesis is, "*How many latent topics exist for each query at a given time?*" The easiest way to know this is to fix the number of topics (e.g. $K = 30$), and then it is possible to optimize parameter K with evaluation data.

However, theoretically, the number of topics should be both query and time dependent. From a topic extraction perspective, the number of mixture components (topics) is unknown *a priori* and is to be inferred from the data. Some existing research studied the number of topics problem by using the Chinese Restaurant Process (Blei, Griffiths, Jordan, & Tenenbaum, 2004; Teh, Jordan, Beal, & Blei, 2006), which is a relatively complex process. In this thesis, to make the process easier, and due to the large variation between different queries, dynamic training corpus size is used to identify the number of topics. We assume that a larger training corpus may incorporate a larger number of topics. So, if r is the ratio between the training corpus size and the number of topics, $K = num_{train} / r$. It is also possible to train r by using the ranking evaluation for this task.

Both methods can be tested in the ranking task with the real-world evaluation.

3.3.3 Community Interest Vector (CIV) ranking

This section and the next explain the methodological stages of community interest based ranking. The central task for the new ranking algorithm is community interest modeling. Mathematical modeling is frequently used with the objective to understand, explain, reason and predict behavior or phenomenon in the real world (Hiemstra, 2001).

The first ranking algorithm introduced is community interest vector (CIV) ranking, stemming from the classical vector space model (Salton, Wong, & Yang, 1975).

3.3.3.1 Community interest vector generation

As mentioned in section 3.3.1, community interest topic space is extracted from real-time user generated blog postings, and, as Figure 3-2 shows, community interest is defined as a distribution of degree or probability of interest in the community interest topic space for a given query at a given time.

For *CIV ranking*, community interest can be used as a dynamic vector, and each component of the

vector represents a (normalized) topic related to the target query, representing the degree of that community currently interested in this topic for the given query. This query interest vector may change in two different ways over time:

1. *Vector space change (community interest space change) – Since each dimension in the vector represents a topic of interest about the target query, a change in the vector space demonstrates that either a brand new interest topic appeared or an existing interest topic faded out.*

2. *Weight change only – This means that the community's interest topics themselves are stable, but the degree of interest (weight) changes over time. In other words, the community's interests shift from one topic to another.*

If we use the query "*Obama*" as an example (as shown in table 3-2), when K = 3, the CIV of "*Obama*" for the 1st of Aug, Oct, and Dec of 2008 may look like the following:

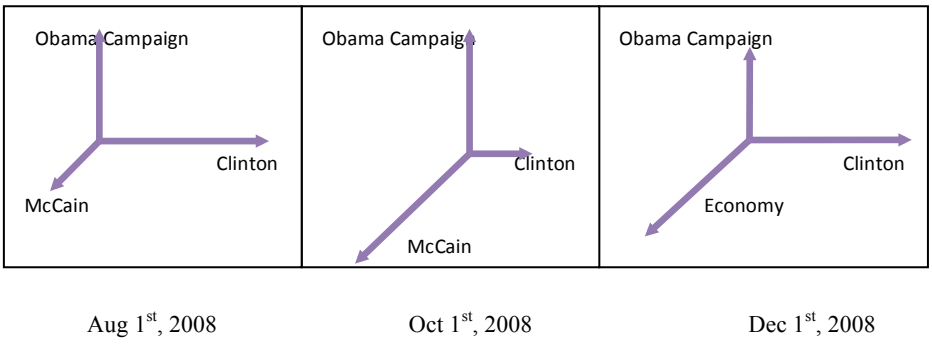

Figure 3-5. Three-dimensional interest topic space and weight change

In August, the community was interested in three different topics about the query "Obama:" 1) Obama's campaign, 2) the relationship between Obama and Clinton, and 3) the relationship between Obama and McCain. The weight of the second topic is larger than the other two, since the community was more interested in this topic compared with the other ones. In October, these topics may exist, but the weights of the first and third topic have increased, while the weight of the second has decreased (weight change). In December, after the election, the third topic is replaced by the "economy," and the weight of each topic changes (interest vector space change).

Vector space dimensionality change can be identified by updating the query dependent training corpus (up-to-date blog postings) with LDA topic modeling, and it is very important to weight and

characterize each topic to capture the topic weight change. The weight of each topic measures the degree of community interest in this topic at the current moment. Overall, there are four different kinds of topics found in the preliminary experiments:

1. *Background topic (stoptopic): the topic covers the very basic background features of the protagonist. Those words, entities and concepts (high probability occurring within topic) can be treated as a protagonist specific stopword list.*

2. *Hot topic: there are two types of hot topics for the community. The first is a topic in which the community is continuously and increasingly focused (but not the background topic, as this kind of topics can be distinguished from longer historical data, and user's interest toward this kind of topics should be increasing over time), and the second is a topic related to breaking news surrounding the query, and which is of sudden great interest in the community.*

3. *Diminishing topic: the topic is no longer popular in the community; the community's interest is shifting to other topic(s).*

4. *Regular topic: The rest of the topics.*

In information retrieval, it is hard to compute the weight of each term in a specific document without modeling the background information of the collection, such as its IDF (inverse document frequency) effects (Spärck Jones, 1972). Similarly, the interest degree of different topics can hardly be measured simply from current training data *CQC* itself. The popularity of the topic (interest based weighting) can be determined by using historical interest data. The reason for using historical interest data is illustrated by the following example: when the number of topics $K = 3$, three topics can be extracted from *CQC*, and the weight of topic z_i could be simply weighted by $W(z_i) = P(z_i|CQC)$, shown in the following chart:

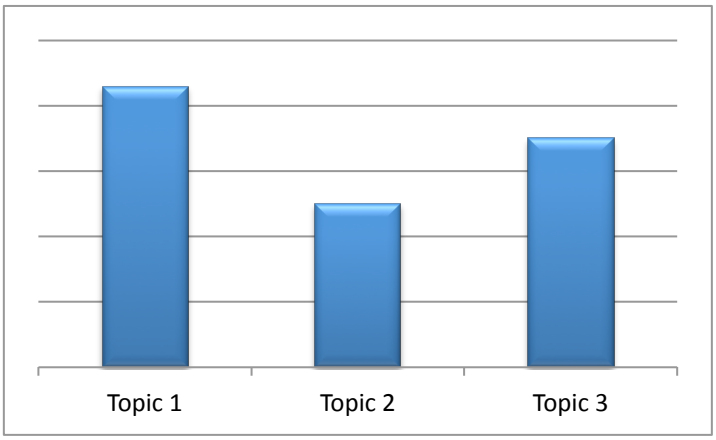

Figure 3-6 an example of topic weights extracted only from *CQC*

From *CQC* itself, it is clear that the Topic 2 score is lower than Topic 1 and Topic 3, which means that Topic 2 is highly likely to be an unpopular topic in this case. However, if historical interest data were taken into consideration, in the past few days or hours, the weight of Topic 2 could be a very small number or almost zero. In this scenario, Topic 2 could be the breaking news about the target query, and it is possible that the community is going to be aware of the importance of this topic; in the next time segment, the community will likely show high interest in this topic. So, Topic 2 could become a hot topic with a higher weight than the other topics. But to know this, we need to use historical data.

To use the historical data, it is necessary to define:

Historical Query-centric Collection (HQC): the historical blog postings collection to represent a snapshot of historical community interest at a given time.

To better represent historical community interest, there could be more than one *HQC* used to weight each topic, just as the following diagram shows:

Figure 3-7. *CQC* (current interest) & *HQCs* (historical interest)

As the above diagram shows, the current interest model of the target query is represented by *CQC*,

and, considering past *n* time segments (e.g. past *n* hours or past *n* days), historical community interest snapshot models can be represented by HQC_1, HQC_2... HQC_n. The weight of topic *i,* given HQC_t, community interested in topic *i* at the *t* snapshot, can be represented by:

$$W_{-t}(z_i) = P(z_i|HQC_t) = \frac{\sum_{Posting_n \epsilon HQC_t} P(z_i|Posting_n)}{N} \qquad (3\text{-}3)$$

In this formula, the weight of topic *i* given HQC_t is represented by $P(z_i|HQC_t)$, which is the sum of the probability of z_i given each posting in HQC_t, $P(z_i|Posting_n)$, divided by the total number *(N)* of postings in HQC_t.

Based on this definition, the input of this algorithm is the current interest (extracted from *CQC*) and historical interest models (extracted from *HQC[n]*), and the output should be the interest vector where each topic is weighted to mirror the degree of real world interest.

The most straightforward method is to compute a list of topic models for each training corpus (*CQC* and *HQCs*) for a specific period of time, and then compute the similarity of those topics, and also weight each current topic for ranking. However, there are two major limitations. First, the computational cost is very high, as we need to train several LDA models and compute feature-topic distribution distance for each topic pair. Second, this is not an accurate way to compute weights when similarity across topics is low.

In order to avoid those limitations, the learned *CQC* topic model is used to infer the topics in the historical protagonist corpuses. The algorithm is as follows:

For each Query

```
Training_CQC_topic_model;                    //k topics
Current_topic_score[k] = ∑document_topic_dist[k]/N;
//compute the CQC topic score by summing each doc-topic distribution in CQC
Compute History_topic_score[n][k]
//Inferring past n days (or hours) topic distribution based on CQC topic model

CIV[k] = 0;   //Community Interest Vector, each score is the weight of the topic
For each topic j
        Compute Mean and Standard_deviation (Std) for history topic score;
        If (Current_topic_score[j] > Mean + Std)
                CIV[j] = b*Current_topic_score[j]*(Current_topic_score[j]/Mean);
                //Hot topic, b is the "bonus" parameter b > 1
        Else If (Current_topic_score[j] < Mean - Std)
                CIV[j] = p*Current_topic_score[j]*(Current_topic_score[j]/Mean);
                //Diminishing topic, p is the "penalize" parameter p < 1
            Else
                CIV[j] = Current_topic_score[j];
                 //Regular topic
              End If
        End If
Next topic
```

Fig 3-8. *CIV building algorithm*

As mentioned above, from the LDA model, two probability distributions were obtained: α_{LDA} - the probability of topic given the posting *P (topic | posting*;--and β_{LDA} and the probability of each feature given the topic *P (feature | topic)*. Based on these distributions, "*Current_topic_score[k]*" is computed by summing the posting vectors from α_{LDA} and dividing by the number of postings. The LDA model is also run against historical data (past *n* time segments, *n* corpuses) and infers the topic distributions α_{LDA} in the historical data. Because the LDA model is built using the *CQC*, the historical postings (*HQC*) can be viewed as unseen data. For each posting, the inference result (LDA based topic representation) is:

$$Posting_i^{LDA} = P(z_1|Posting_i), P(z_2|Posting_i), \dots P(z_k|Posting_i) \qquad (3\text{-}4)$$

which indicates the probability of each specific topic in the unseen document from the current perspective. For each past day or hour, by summing these topic probability vectors together, we can obtain a "*History_topic_score[i][k]*," which reflects, from the current viewpoint (topic model), the probability that in the past *i*th time segment (day or hour) ago, the community (represented by the HQC_i) is interested in topic *k*. By comparing the mean and the standard deviation of specific topics' scores for a window of past *n* time segments, we can decide if the topic is a "hot topic," "diminishing topic," or "regular topic," as shown in the algorithm:

$$CIV[j] = \begin{cases} b \cdot current_{topic_{score[j]}} \cdot \dfrac{current\_topic\_score[j]}{average\left(history_{topic_{score[i][j]}}\right)} & \dots\dots hot\ topic \\ p \cdot current\_topic\_score[j] \cdot \dfrac{current\_topic\_score[j]}{average(history\_topic\_score[i][j])} & \dots diminishing\ topic \\ current\_topic\_score[j] & \dots regular\ topic \\ p' \cdot current\_topic\_score[j] & \dots background\ topic \end{cases} \quad (3\text{-}5)$$

In above formula, the popularity of topic j in CIV is calculated in four different categories, ordered by hot topic, diminishing topic, regular topic and background topics (from top to bottom). The hot-topic and diminishing-topic CIV scores were adjusted by the change rate of the current topic score and the mean of the historical topic scores; a bonus parameter (b, b>1) and penalty parameter (p, p<1) were used in the algorithm to update the topic weight. In the preliminary experiments (Liu & Brzeski, 2009), b = 1.2, while p = 0.8. Because the topic category is identified by mean and standard deviation of the history interest scores, the change rate of a hot-topic is always > 1 and the change rate of a diminishing-topic is always < 1. In a preliminary experiment, it was found that some topics' mean probability score was significantly larger than all other topics scores (at least 5 times larger), and these topics were defined as the "background topics." Because they are background topics, these topics' weights were penalized by p' (penalty parameter of background-topic, current setting of p' = 0.2). The background topic is mainly composed of a list of general and domain specific "stopwords" (for instance query = "Obama," the query specific stopwords can be "Obama" "US" and "president"). Even though the background topics' weights are large in all the corpora, these topics are harmful for community interest based ranking.

The following diagrams are examples of CIV topic weighting. "Obama" is used as the example,

and the experiment time (current community interest model) is *Nov 5th 2008 14:00*, one day after the 44th presidential election. The training corpus is Yahoo! Buzz postings; and, the current query training corpus size is *1,491* (user generated postings), and a list of historical training Buzz blog corpuses were used (each more than 1,000 postings). We show the highest weighted "Hot topic," (Figure 3-9) which can be summarized as "Obama wins the election with a new record," whose top features are "Barack_Obama," "Election," "African_American," "victory," "Victory_Records" and "first_black_president." We also show the lowest weighted "Diminishing topic," (Figure 3-10) which can be characterized as "Sarah Palin and Hillary Clinton," with top features like "Sarah Palin," "sarah," "palin," "Hillary Clinton," "newsweek," and "club."

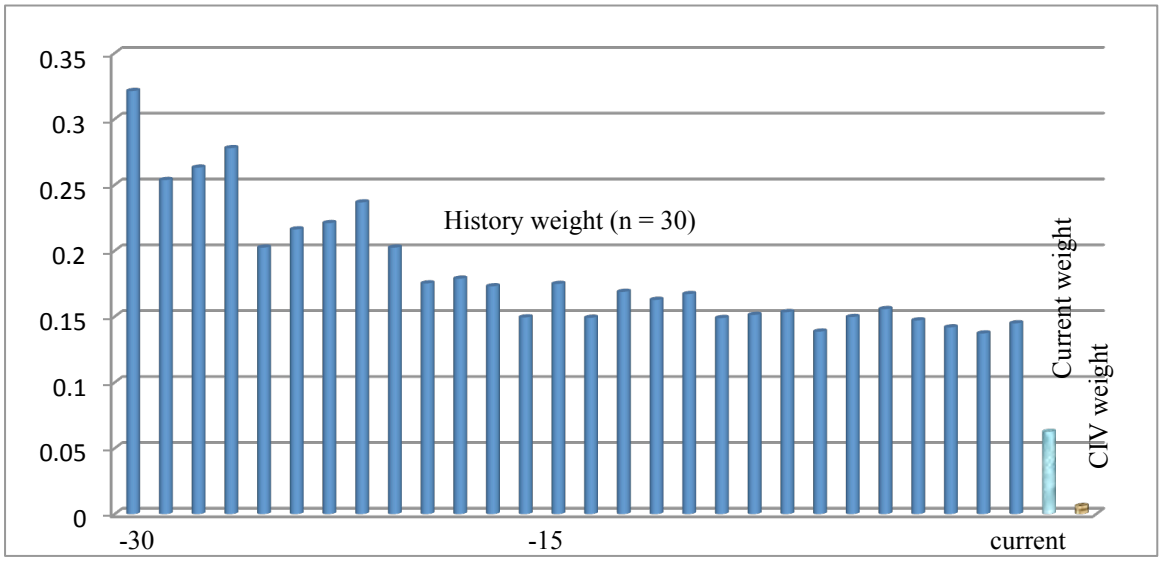

Figure 3-9. Nov 5th, Diminishing topic: "Sarah Palin & Hillary Clinton"

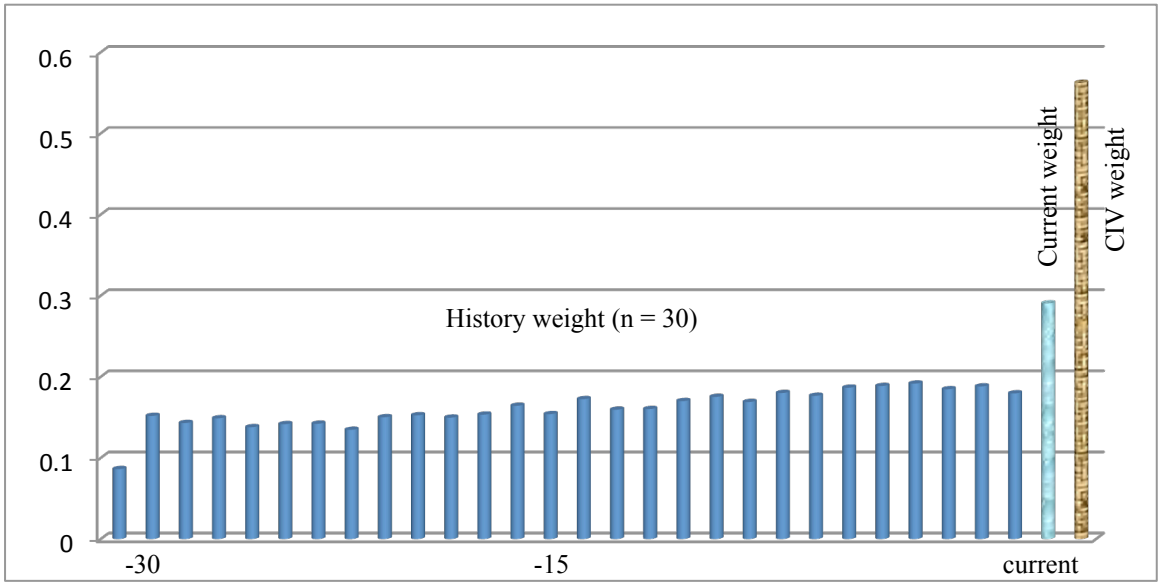

Figure 3-10. Nov 5th, Hot topic: "Obama win president election"

In the example (Figures 3-9 and 3-10), the community interest topic space was trained using the current "Nov 5th" corpus, *CQC*, about "*Obama*," and then it was used to infer topic distributions, $P(z_i|HQC_t)$, in the past 30 days (from Oct 5th to Nov 4th, *HQC[30]*). By computing the mean and the standard deviation of the topic probability scores, hot and diminishing topics can be identified by their final weights in the *CIV*. In the diagram, the second bar on the right is the initial (current) topic weight, and the last bar on the right is the final adjusted weight of the topic in the *CIV*.

3.3.3.2 CIV based ranking

When a query is equal to a *CIV* protagonist, the current Community Interest Vector can be used to bias the ranking result. For any given retrieved document collection $R = (doc_1, doc_2, \ldots\ldots doc_n)$, based on the topic model (α_{LDA}, topic-feature distribution), each document in the search results can be represented with a topic distribution vector, $TV(doc_m)$, in the community interest topic space by inferring the topic model as mentioned earlier. Because the topic vector of each retrieved document in the search results is in the same vector space as *CIV*, we can compute the final document interest ranking score by cosine vector similarity:

$$ranking\_score(doc_x) = Sim(CIV, TV(doc_x)) = \frac{\sum_{i=1}^{n} CIV(topic_i) \cdot TV(doc_x, topic_i)}{\sqrt{\sum_{i=1}^{n} CIV(topic_i)^2} \cdot \sqrt{\sum_{i=1}^{n} TV(doc_x, topic_i)^2}} \quad (3\text{-}6)$$

Since the *CIV* represents the community's current interest with respect to each query, the final ranking score can be viewed as a pseudo-voting from a community interest perspective, where the user oriented text data serves as a proxy for the votes. Thus, the ranking score can represent $P_{community}(interest|doc_i)$, the probability that a community is interested in a given retrieved document.

3.3.4 Community Interest Language Model (CILM) ranking

The language model is another effective probability ranking model first introduced by (Ponte & Croft, 1998b). In this retrieval model, the documents in the retrieved result are ranked by query likelihood scores. The query likelihood method not only performs well empirically, but also has a

solid relevance-based foundation (Zhai, 2008). The language model ranking mechanism is described as follows:

1. *Estimate the language model for each document in the collection*
2. *Rank the documents by the likelihood of query given the estimated language model,* $P(query|\theta_{doc})$

Based on this definition, both query and document are assumed to be samples of words drawn from the (query and document) language models. One of the key problems for the language model is how to estimate θ_{doc} given a document and collection; Zhai (2002), Zhai and Lafferty (2004) found that the retrieval and ranking performance is sensitive to smoothing methods.

One assumption behind language model ranking and query likelihood is that if the document author is represented by the document language model: What is the likelihood that she (document language model) proposes the target query in the retrieval system?

The objective of this study is to rank the retrieved documents for a given current community interest model. If community interest is defined as a dynamic *community interest language model* θ_{CI}, the assumption can be made that each term or (latent) topic in a retrieved document might be generated from θ_{CI}, which is the document likelihood given the interest language model. If the retrieved document is ranked partially by this probability score, the language model ranking function can be written by the linear combination of query likelihood given the document language model θ_{doc}, and the document likelihood, given interest language model θ_{CI}:

$$(1-\varphi)\, P(query|\theta_{doc}) + \varphi \cdot P(doc|\, \theta_{CI}) \qquad (3\text{-}7)$$

Compared with the query, the document is a much larger sample for estimating what the author had in mind, which enables the ranking algorithm to judge the user's interest (in this document) by leveraging θ_{CI}. As defined in the first chapter, the gap between the user information need and the query string (α), is somehow unpredictable. However, since documents tend to be much longer, it is highly likely that the document interest probability can be estimated from the query perspective by leveraging an interest language model θ_{CI}, since each term or each topic in the document has a different probability of interest to them.

The first part of this formula is the classical language model (query likelihood) score, while the second part is the community interest based document likelihood score, which we will focus on in the next section. Also discussed will be the estimation of θ_{CI}.

From a language perspective, the most straightforward estimator of θ_{doc} is the maximum likelihood based on the document itself, and smoothing the ML estimator is critically important in the language model for two reasons (Zhai, 2008):

- *First, it addresses the data sparseness problem. As a document is only a very small sample, the probability P (q_i | Doc) could be zero for those unseen words in the document (Zhai & Lafferty, 2004).*
- *Second, smoothing helps to model the noisy (non-discriminative) words in the query.*

For community interest computation in this thesis, the focus is on community interest topic space, and the data sparseness problem will not matter because, in most cases, the inferred $probability$ is a non-zero number. However, modeling noisy topics is critically important for ranking.

As mentioned in the last section, the topics in the community interest topic space include background topic, hot topic, and diminishing topic, and for the CIV algorithm, trend analysis is employed to classify the topics for weighting and ranking. In the language model, compared with TF-IDF weighting parameters, the smoothing parameter is more meaningful from the point of view of statistical estimation and it models the background language model by using $P(term|C)$ (Zhai, 2008).

As formula 3-7 shows, community interest is composed of two different components: current interest (extracted from *CQC*) and historical interest (extracted from a list of *HQCs*). Historical community interest can help us weight topics. Similarly, in the community interest language model, historical interest can help us model the noise in the community interest topic space for ranking as the smoothing parameter, such as the following:

$$\log P(doc|\theta_{CI}) = \log \frac{P(doc|\theta_{CI-now})}{\alpha_{CI} \cdot P(doc|\theta_{CI-history})} \qquad (3\text{-}8)$$

In the above formula, the probability that θ_{CI} generates the document can be computed by dividing the probability of a document given θ_{CI-now} by the probability of a document given

$\theta_{CI-history}$. So, if a document gets a high ranking score, it should have a high current interest probability score (document topics of interest to the current community) with a low historical interest probability score (document topics not of interest to the historical community).

α_{CI} in the formula controls the amount of smoothing to estimate the interest language model θ_{CI}. In the classical language model, parameter α_D is also used as a document-dependent constant (Zhai & Lafferty, 2004) and it can be used to normalize for document length.

As stated earlier, each candidate document in the retrieved result collection can be represented in the community topic space and $P(doc|\theta_{CI-now})$ can be calculated by using the LDA topic model of *CQC*. The remaining problem is how to calculate $P(doc|\theta_{CI-history})$ by using a list of *HQCs* (historical community interest snapshots).

Intuitively, historical community interest should be a decay function, as the more recent community interest snapshots should have a larger contribution when comparing the interest model with old snapshots. Based on this hypothesis, the following recursive function is employed to define $\theta_{CI-history}$ with the user interest decay parameter π. :

$$\theta_{CI-1} = \theta_1 \;\; ; \quad \theta_{CI-k} = \frac{\theta_k + \pi \cdot \theta_{CI-(k-1)}}{1+\pi} \qquad (3\text{-}9)$$

In the formulas, θ_{CI-n} is the oldest community interest snapshot, and it is estimated by the inferred topic probability distribution θ_n (from the LDA model). For any θ_{CI-k}, it is defined (normalized) by θ_k and $\theta_{CI-(k-1)}$ with an interest decay parameter π. Based on this definition, finally, the $\theta_{CI-history}$ is:

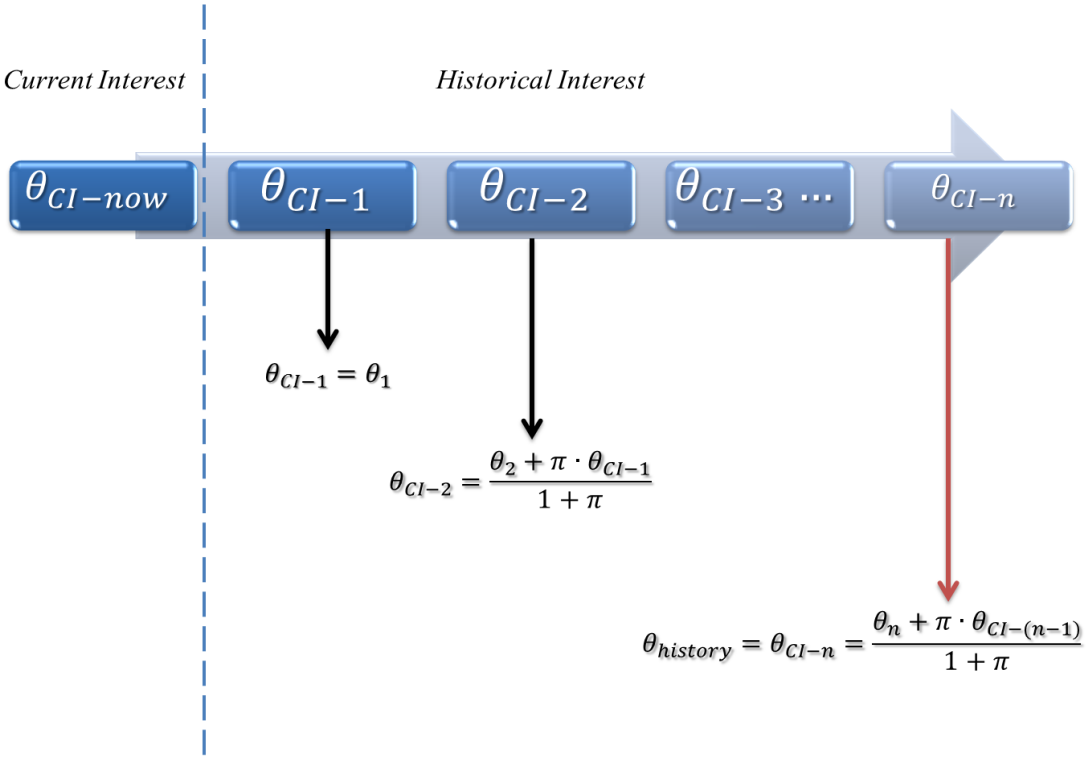

Figure 3-11. Historical interest modeling and decay function

So, $\theta_{CI-n} = \theta_{history}$ is composed by $\theta_1, \theta_2, \theta_3 \ldots \theta_n$ (all the historical HQC snapshots). While θ_n is the oldest snapshot (the best representation of $\theta_{history}$, and n is a parameter will be trained later this chapter), the representability of $\theta_{n-1}, \theta_{n-2}, \theta_{n-3} \ldots \theta_1$ decreases, and the decay parameter is $\frac{\pi}{1+\pi}$. When π is small, the decay speed is fast, when π is large, the decay speed is slow. For instance, when π = 0.1, the decay parameter is 0.0909; when π = 10, the decay parameter is 0.909. Finally, the retrieved documents are ranked by the real time community interest generated probability scores. In this research, π is a key parameter that needs to be trained. Intuitively, in different types of searches, i.e. news search or web search, decay functions should be different.

3.4 Evaluation

Evaluation is important for all empirical information retrieval studies, and in this study, evaluation is more challenging due to the following reasons:

1. *An interest based ranking method is very dynamic, so a traditional static relevance based evaluation cannot be used, as a user's interest toward each query and document may*

change over time and their interest judgment may also change from time to time.

2. *Document (retrieved result) interest should not be simply classified into interest / non-interest. Instead, the documents should be ranked by the degree of interest.*

3. *Some existing standard ranking test evaluation resources, such as used in TREC, cannot be used directly in this experiment, because they are not dynamic evaluation collections.*

In the next section, the evaluation method as well as the preliminary experiment and evaluation result will be discussed.

3.4.1 Evaluation method

In modern information retrieval systems, retrieval and ranking algorithms tend to overwhelm system users with a very large number of retrieved and ranked results. This requires evaluation methodology to measure the top results with a document cut-off value (DCV). For this study, Normalized Discount Cumulative Gain method is used:

Normalized Discount Cumulative Gain (NDCG) (Järvelin & Kekäläinen, 2002) works when user graded relevance data is available, which estimates the cumulative relevance gain the user receives by examining the retrieval result up to a given rank on the list. NDCG is based on two facts: first, highly relevant documents are more valuable than marginally relevant documents (graded relevance judgment), and second, the lower the ranked position of a relevant document, the less valuable it is for the user. A ranked vector *V* of results [$label(v_{1)}$, $label(v_{2)}$... $label(v_k)$] can be generated for each query q where each item in the vector is the judgment of degree of relevance (e.g. 0 is not relevant and 5 is perfect relevant). With this vector, calculation of the Discount Cumulative Gain (DCG) is possible:

$$DCG@k(V) = \sum_{i=1}^{k} \frac{1}{log_2(1+i)} (2^{label(v_i)} - 1) \qquad (3\text{-}10)$$

The normalized DCG (NDCG) of *V* is defined as the DCG vector divided by the ideal permutation of *V*.

NDCG is a good method to evaluate web based ranking algorithms, and it can be integrated with significance tests to show the robustness of a new ranking algorithm.

For this thesis, the most important contribution is to capture the time-dependent community interest for ranking. As a result, it is necessary to conduct a real-world evaluation based on selected queries over a period of time. Instead of using "relevance judgment," the focus is on a user's judgment of the "degree of interest" for each top ranked retrieved document. In the evaluation system, users access the top ranked documents and judge each of them with a four-point scale for interest assessment:

> *Interesting and Hot (3):* Very interesting retrieved result for the given query at the moment. The user is highly likely to read the document immediately and the content of the document is relevant to the target query.
>
> *Interesting (2):* The content of the document is interesting to the target query, but not as strongly as "Interesting and Hot".
>
> *Just OK (1):* The content of the document is somehow interesting to the target query, but the user may not willing to spend a long time reading it, or the topic of the document is not up-to-date.
>
> *Not interesting or not relevant (0):* The content of the document is tedious even though the content may be relevant to the query or the content of the document does not match the query at all.

When the user evaluation matrix is made available, NDCG is used to judge the effectiveness of the new ranking algorithm by comparing it with a base line from existing ranking algorithms.

As stated in the first chapter, there are two kinds of ranking algorithms, relevance based ranking and popularity based ranking. Most popularity based ranking algorithms employ statistical user data, such as clickthrough, dwell time and universal hyperlinks, however, those data are hard to access or collect and, as a result, it is hard to implement popularity based ranking algorithms directly as a baseline for comparison.

The compromise is to compare the new ranking algorithm with existing search engines' ranking results, which has normally relied on a complex combination of popularity based ranking algorithms.

The following methods and parameters were evaluated:

1. *Compared NDCG of CIV and CILM ranking algorithms with relevance based ranking method (BM25, vector space model and language model with different smoothing methods) and search engines' ranking result (Google[3] and Yahoo[4]). The results were evaluated with statistical significance tests.*

2. *Community interest vector parameter setting was evaluated. Both the number of topics setting, and the length of time for trend analysis was evaluated.*

3. *Community interest language model parameter setting was evaluated. To carry out the evaluation, first, the number of topics setting, and then the length of time for historical smoothing along with decay parameter π was trained and optimized.*

A real-world evaluation was launched using Amazon Mechanical Turk (MTurk)("Amazon Mechanical Turk, https://www.mturk.com/,"). MTurk is a human based artificial artificial intelligence system, also known as a social computing system, which can accomplish certain tasks by means of collecting human judgments. MTurk has been used in information retrieval and text mining related research (Alonso, Rose, & Stewart, 2008; Dakka, Gravano, & Ipeirotis, 2008; Evans & Chi, 2008; Liu, Bian, & Agichtein, 2008; Yang, et al., 2009). These studies demonstrated that MTurk is a promising method for evaluation.

From the developer and researcher (as requester) side, tasks can be submitted through a web service or API to the Amazon Mechanical Turk. From the user (as worker or Turker) side, users can sign in at the MTurk web site and work on selected project(s) while receiving payment for their work. According to (Alonso, et al., 2008), there are over 200,000 registered workers from over 100 countries, and millions of tasks have been completed.

In this thesis, the following steps were used for evaluation:

1. *A list of hot queries was identified from recent query logs.*

[3] www.Google.com

[4] www.Yahoo.com

2. *For each query, a list of recent blogs (for at least 7 days) was collected. If a query's daily average number of blog postings was larger than* num_{min}*, this query was used for evaluation.*

3. *A list of top ranked retrieved results (e.g. top 10 documents) was saved (with rank order information). The saved results were re-ranked with community interest ranking algorithms (CIV and CILM with most recent, e.g. past 1 hour, community interest model) and relevance based ranking algorithms (vector space model, language model and BM25).*

4. *For each candidate query, the saved results were uploaded to Amazon Mechanical Turk in a random order; users evaluated the degree of interest of each result for the given query within 5 hours (after the result was uploaded).*

5. *Based on the evaluation result, the performance of different ranking algorithms (relevance, popularity and interest based ranking algorithms) was compared, and different parameter settings were compared by using NDCG.*

6. *Statistical significance tests were used to test the significance of the results.*

For this evaluation setup, when there are *m* queries and top *n* results retrieved every day, and the experiment lasts for *t* days, the total evaluated number is $m \times n \times t$. The evaluation stability issue was studied by Buckley and Voorhees (2000). According to Spärck Jones and Rijsbergen (1976), 75 requests need to be judged. The TREC program committee used 25 topics as the minimum while 50 topics is the norm (Voorhees & Harman, 2000). As a consequence, a reasonable number of queries were evaluated to test this thesis' hypothesis. Another limitation for this research is we only re-rank the top 10 retrieved results from different search engines, because we have limited resources (i.e. to pay Amazon Turkers). A more comprehensive evaluation is expected, which will be mentioned in Chapter 6.

The detailed experiment steps will be described in next chapter, such as the number of queries, method of data collection, inter-coder reliability.

Chapter 4: Experiment

4.1 Introduction

This chapter describes the experiments that were run to test the performance of the innovative ranking method with respect to real-time community interest, the evaluation process, and the results. The next chapter will interpret the experimental results and their significance to information retrieval.

Section 4.2 describes the data collection process and the use of the data, section 4.3 describes in detail the experiments and evaluation process, and section 4.4 presents the evaluation result. The interest based ranking algorithms will be evaluated by comparing their ranking results with those of a variety relevance based ranking methods and search engines' ranking results.

4.2 Data collection

The major goal of this experiment was to test the real-time interest based ranking performance of a test query collection, in part, through the use of a large amount of user-generated real-time interest judgments for the top retrieved results. The interest model was trained by a set of time-sensitive blog postings.

In order to achieve this goal, four types of data were collected: the test queries, the blog postings, daily or weekly rankings from popular search engines, and user oriented interest judgments. In this section, each kind of data will be described.

4.2.1 Query collection

As mentioned in the previous chapter, the TREC program committee used 25 topics as a minimum for evaluating retrieval results while 50 topics is the norm (E. M. Voorhees & Harman, 2000). For this experiment, 50 test queries were selected for experimenting and evaluation. The selected queries, the popular ones during the time period of the evaluation, were chosen for two reasons. First, popularity ensures that a reasonable amount of real time blog postings can be collected to train the interest model for interest based ranking; and, second, users understand the queries well

and they can easily provide their relevance or interest judgments for the retrieved results given a target query. In this experiment, a list of popular queries was identified by using “Google Insights for Search” (http://www.google.com/insights/search/), which was also used to collect the query terms.

“Google Insights for Search” was used in two ways:

1. The top searches (queries and query terms) and top rising searches (queries that were analyzed as rapidly gaining popularity) for Google news search or Google web search for the past fixed period of time, i.e. past 7 days or past 30 days are identified by the utility. The results can be filtered by location and category. The following diagram shows the top 10 web search terms and rising search terms for location United States for the past 30 days, selecting all categories. The numbers on the graph are the normalized scores (0 to 100) that reflect how many searches have been done for a particular term, relative to the total number of searches done on Google over time.

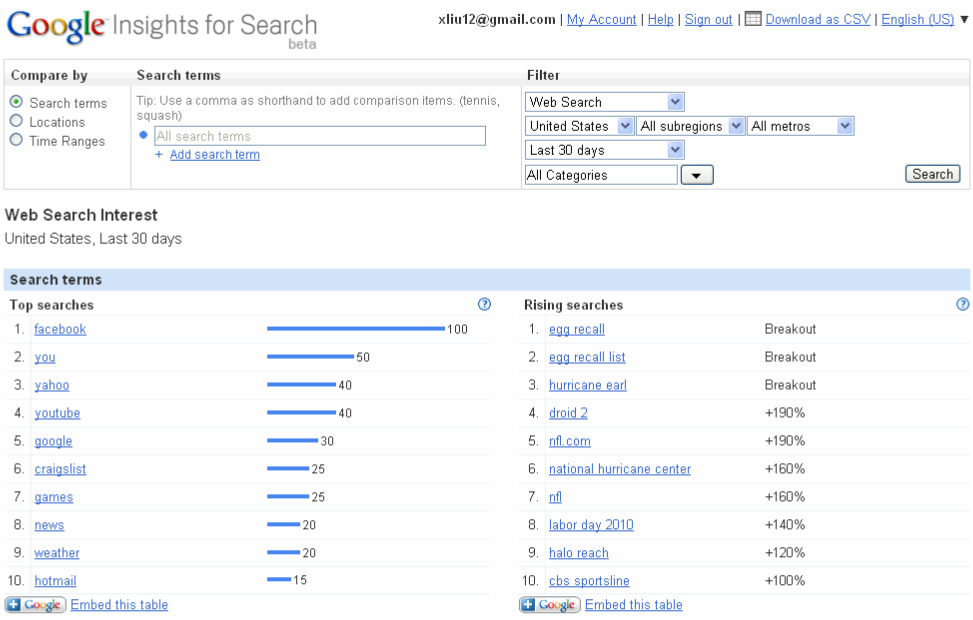

Figure 4-1. Top search index from Google Insights

2. The popularity hypothesis of a particular query or query terms given a period of time and a category can be reviewed. For instance, the following diagram shows the normalized popularity score of the query terms, “Obama” and “BP,” for the past 30 days in Google News Search for all categories. Meanwhile, the search popularity trend also can be seen in the diagram.

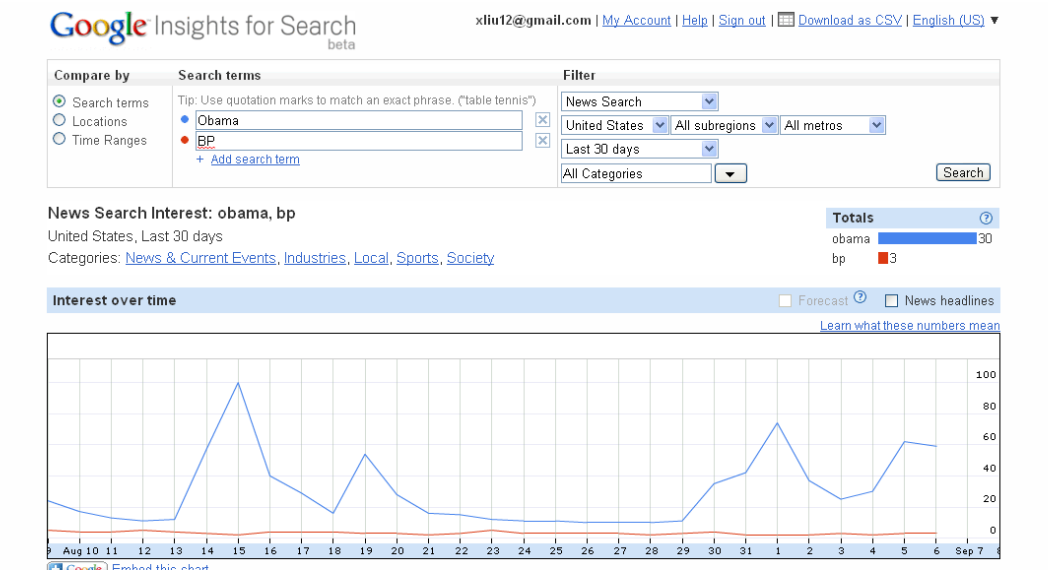

Figure 4-2. Test query popularity with Google Insights

50 test queries were selected by using Google Insights. Some popular queries were ignored because they do not reflect dynamic information needs and thus users' interest toward these queries may be stable. Examples include "YouTube", "Facebook" and "Google", which are used primarily to identify a list of URLs not topics of interest. The selected queries can be categorized into the following classes:

- *Long-lasting popular news stories (14 queries)* such as "Obama", "Palin" and "Dow Jones". Users' or community interest toward these queries may or may not change based on new events.
- *Recent popular news stories (17 queries)*. These are queries such as "oil spill", "wikileaks" and "NBA Trade". Users' or community interest toward these queries may change dynamically in a short amount of time.
- *World cup (soccer) (19 queries) related queries.* Examples include 2010 South Africa World Cup (from 11 June to 11 July 2010), "World Cup Netherlands", "World Cup Final" and "World Cup Championship." Users' or community interest with respect to these queries changed a great deal (i.e. hourly) during the games.

4.2.2 Blog posting collection

In this thesis, real time global community interest is extracted from chronological *user oriented blog posting data.* Google blog search[5] was used to collect this query centric training data over a period of four weeks. Because most of the queries were popular queries during the time of the evaluation, a considerable number of blog postings were available and collected for interest model training. For each query, an average of 28,773 postings (covering, on average, four weeks) was indexed for training.

Queries that didn't result in a minimum of 200 postings/day averaged over the collection time period were ignored for the evaluation experiments. The blog postings were indexed by content (title and content), URL, time (posting generation date and hour) and the target query.

A limitation of using Google Blog Search is that the retrieved result could be relevant to the query, but the query is not necessarily the pseudo protagonist of the retrieved blog postings. For instance, a posting that mentioned "Obama", might be statistically relevant, but not *about* "Obama," and this could threaten the precision of the interest model for ranking.

4.2.3 Ranking results collection

As mentioned in the last section, for each query, blog postings were collected for 4 weeks after the start of the experiment. The first 10 days' postings were used only for historical community interest inferencing (i.e. smoothing or trend analysis). From day 11, the ranking results from search engines were collected for evaluation purposes. Because the goal was to test the effectiveness of the innovative interest based ranking algorithm, it was important to set up the re-ranking tasks across different kinds of search engines. In this thesis' experiments, two types of search were used – news search and web search from both Google and Yahoo, two of the most popular search providers. It was expected that news ranking results would be more dynamic than web ranking. Please note, even though this evaluation is a re-ranking task, community interesting ranking algorithm can be used as a ranking algorithm, i.e. rank the retrieved documents by probability of interest.

The top 10 ranked retrieval results were indexed for every query at 2:00PM. The ranking position

[5] http://blogsearch.google.com/

of each news document was stored along with HTML content and experiment date. This process lasted on average 10 days, depending on whether enough user judgment data was collected. Similarly, the top 10 ranked retrieved results for each query from Google and Yahoo web search were indexed twice a week. Since the ranking results from web search are more stable than news search, only an average of 2 weeks' worth of rankings were kept.

4.2.4 User judgments collection

In order to test the effectiveness of the innovative ranking algorithm, the community interest based ranking result was evaluated against search engine rankings as well as against a variety of relevance rankings based on different ranking algorithms. The differing algorithms were run on the retrieved results from the daily top 10 ranked documents for both news and web search. In this section, we describe the process of collecting users' real-time interest judgments on the top ranked documents.

Since users' or community interest can be dynamic and the ranking results can change based on the interest judgment, we need to collect users' real time interest judgments, and use the averaged user interest judgment as the gold standard for ranking algorithm evaluation. Immediately after the rankings list from each search engine is indexed for each query, an evaluation task is setup by using Amazon Turk, as shown in the following diagram:

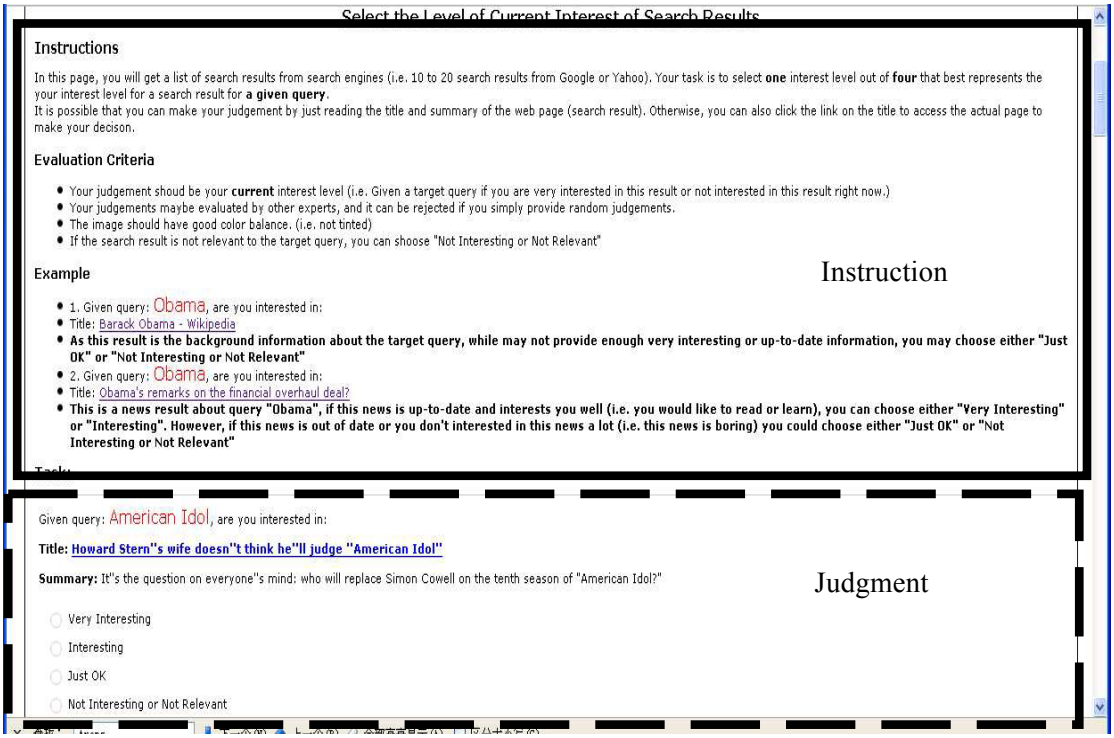

Figure 4-3. Amazon Turk evaluation page

With instruction and examples, Amazon Turk users (known as turkers) are expected to provide the real time interest and relevance judgments for retrieved documents given a query. As mentioned in the last section, the ranking information on the top retrieved documents is collected from different search engines at 2PM every day for news search, or twice a week for web search. At the same time, the collected retrieved documents will be sent to Amazon Turk in a *random order*. For each query, for one day, the documents are shown on one evaluation page, which is called an Amazon Turk HIT. To minimize potential individual bias, up to 5 different turkers worked on each HIT. The task needed to be completed within five hours after the HIT was created, as the user's interest could change too much after that. For any HIT, if there were fewer than three turkers working on it, the HIT was deleted from the database and wasn't used for evaluation.

In this evaluation, a reasonable and (researcher) affordable compensation rate is important to attract turkers to work on the HIT within a limited time. In order to find enough turkers to accomplish the task, we tried different compensation rates starting from $0.01. Not surprisingly, a higher compensation rate is very helpful to attract more turkers, and we also found, when the compensation rate was raised to $0.11, in most cases, we can accomplish the real-time evaluation task within three hours. We assume the reason is that some turkers look for HIT jobs by searching

by price, and sometimes, they filter out HITs lower or equal to $0.10. At the same time, to find a larger number of turkers, we provide a list of keywords to describe this evaluation task, such like "evaluation", "Google", and "easy task".

In preparation for this task, turkers were given three piece of information:

1. The basic instruction needed to finish this evaluation task:

> In this page, you will get a list of search results from search engines (i.e. 10 to 20 search results from Google or Yahoo). Your task is to select one interest level out of four that best represents your interest level for a search result for a given query.
> It is possible that you can make your judgment by just reading the title and summary of the web page (search result). Otherwise, you can also click the link on the title to access the actual page to make your decision.

2. The criteria by which a turker's judgment was evaluated. A turker's work could be rejected if their judgment quality was found to be low.

> Your judgment should be your current interest level (i.e. given a target query if you are very interested in this result or not interested in this result right now.)
>
> Your judgments maybe evaluated by other experts and it can be rejected if you simply provide random judgments.
> If the search result is not relevant to the target query, you can choose "Not Interesting or Not Relevant"

3. Examples given to turkers to facilitate the judgment process. Turkers were encouraged to provide their own judgments.

> 1. Given query: ***Obama***, are you interested in:
>
> Title: Barack Obama - Wikipedia
>
> As this result is the background information about the target query, while may not provide enough very interesting or up-to-date information, you may choose either "Just OK" or "Not Interesting or Not Relevant"
>
> 2. Given query: ***Obama***, are you interested in:
>
> Title: Obama's remarks on the financial overhaul deal?
>
> This is a news result about query "Obama", if this news is up-to-date and interests you well (i.e. you would like to read or learn), you can choose either "Very Interesting" or "Interesting". However, if this news is out of date or you don't interested in this news a lot (i.e. this news is boring) you could choose either "Just OK" or "Not Interesting or Not Relevant"

In the judgment process, for each retrieved document, the HIT showed the following information: the target query, highlighted in a different size and color; the title with a hyperlink to the actual page in a different window; and a snippet from the search engine (Google or Yahoo). Turkers chose one interest level from four listed choices as following:

Given query: ***unemployment***, are you interested in:

Title: Unemployment Compensation

Summary: Welcome to the Unemployment Compensation Program in the Florida Agency for Workforce Innovation. Unemployment insurance provides temporary wage replacement ...

- Very Interesting
- Interesting
- Just OK
- Not Interesting or Not Relevant

Turkers were pre-selected using the following criteria: 1) they reside in the United States and are proficient in English; 2) the turker's pre-study abandonment rate was lower than 81%; and 3) the turker's pre-study approval rate was larger than 50%. While 2) and 3) are popular Amazon Turk criteria, 50% pre-study approval rate is not high. However, this is a real-time evaluation task, and we need a reasonable number of turkers work on each HIT in a short amount of time. So, we a relatively loose pre-study approval rate in order to attract a sufficiently large pool of turkers.

There were 388 distinct turkers that participated in this evaluation. A total of 48,570 judgments were collected over different query document pairs during the study. The size of the final query dataset was 45. Not enough evaluation judgment data was collected for 5 additional queries and they were ignored for the evaluation task.

In order to test the reliability of the turkers' judgment data, we need to compute the inter-coder reliability. We found a large percentage of turkers only worked on one or two HITs and then left. So, we defined the reliability as:

$$\text{reliability rate} = \left(\sum_{\text{for each HIT}} \frac{\text{Max(num of judgments on one interest level)}}{\text{num of turkers working on the HIT}} \right) / \text{num of HITs}$$

In the above formula, the reliability is defined as the average agree rate for all the evaluation HITs. For each HIT, $\text{Max(num of judgments on one interest level)}$ is the number of majority turkers votes a specific interest level. As mentioned earlier, in most cases, the number of turkers working on each HIT is 5. Based on the formula, the reliability rate of this evaluation is 0.548, which means most of turkers agree with each other when working on the same task, even if they were encouraged to provide the judgments given their own opinion. This, in part, ensures that the judgment data collected by Amazon Turk is reliable.

4.3 Experiment setup

4.3.1 Experiment design

In order to test the performance of the interest ranking algorithms, a list of relevance ranking algorithms and search engine ranking results was used as the baseline for comparison. The experiment was designed as follows:

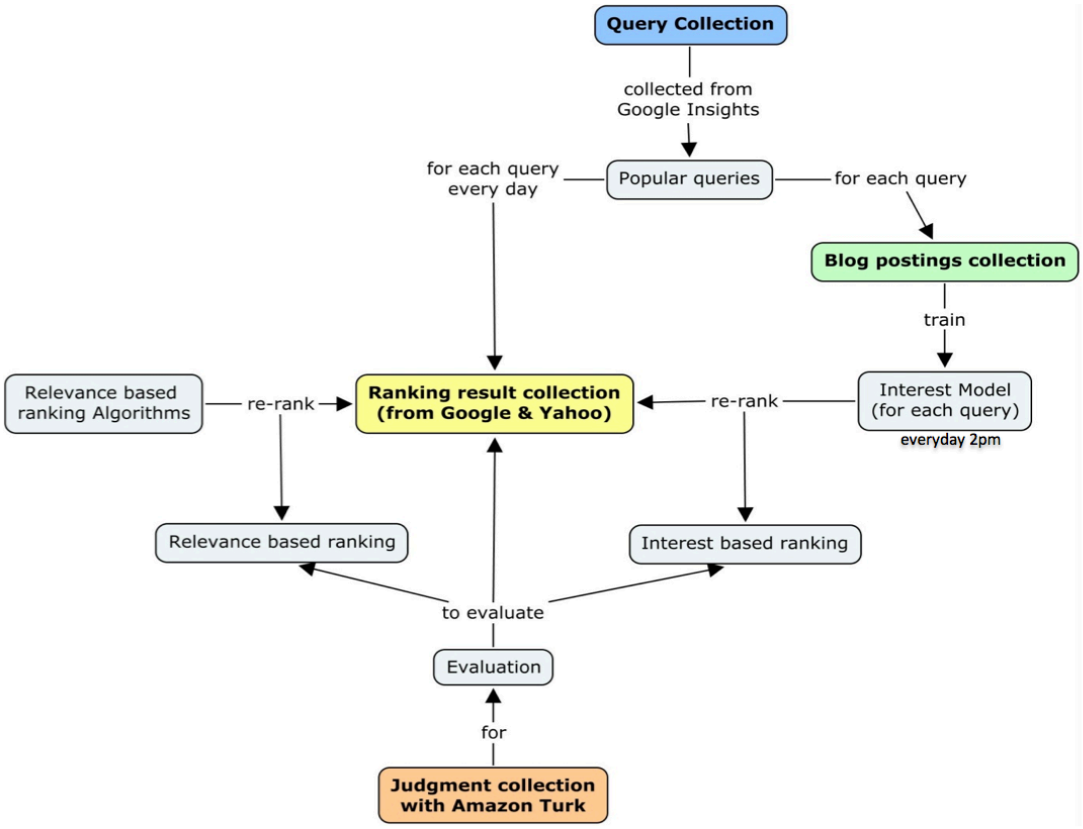

Figure 4-4. Experiment workflow

First, the 45 popular test queries collected from Google Insight Search were used to collect the blog postings for a month to train the real time interest model and ranking information along with the top retrieved results from different search engines.

Second, the top retrieved documents were evaluated by Amazon Turk users for their degree of interest based on the target given query.

oh the trained parameters were applied to the full dataset to build interest models for ranking. News search and web search have different characteristics, and the parameter training processes were run separately for each.

Last, community interest ranking algorithms' (CIV and CILM) performance (NDCG@3, NDCG@5 and NDCG@10) was tested against relevance ranking algorithms (vector space, BM25, language model with linear smoothing, language model with Dirichlet smoothing and language model with two stage smoothing) and against search provider's rankings (Google or Yahoo ranking).

4.3.2 Algorithm parameter training

Using the average interest level of turkers' real time judgments (5 judgments for each document and query pair), NDCG was used to train the parameters of CIV and CILM algorithms.

While different kinds of parameters could be trained for this research, such as LDA related parameters, prior research, such as Wei and Croft (2006), have shown that some parameter setting is not sensitive to the ranking performance. As a result the training was limited to three kinds of parameters: number of topics, historical inference or smoothing length, and decay speed for history smoothing CILM. Again, news search and web search have different natures, and their training processes were separated.

In this thesis, given a small sample of testing query collection, cross folder validation was employed to train and test the ranking performance for both news and web experiments. The 45 queries were randomly separated into three groups, and leave-one-out testing was used to train and test. For instance, in the first round, group 1 and 2 (30 queries) were used for training, and the optimized parameter setting was used to test the group 3 ranking performance.

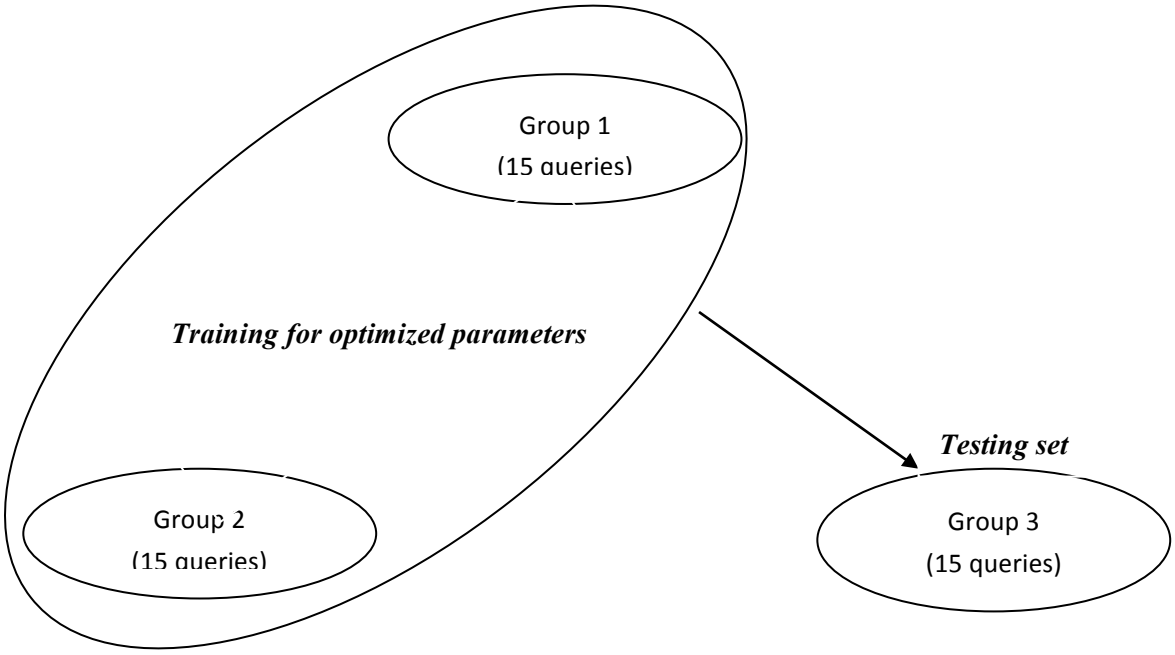

Figure 4-5. (Cross Folder) Training news or web ranking parameters (first round)

For the community interest vector (CIV) algorithm, the number of topics (t_{CIV}) and history

inference days (n_{CIV}) were trained together and the optimized t_{CIV} and n_{CIV} combination were identified.

For the community interest language model (CILM) algorithm, the number of topics (t_{CILM}) is first identified from the training data with a fixed length of smoothing n_{CILM} and decay factor π. Then n_{CILM} and the decay factor (π) was trained given the optimized number of topics.

In this thesis, the default units of n_{CIV} and n_{CILM} are "day", a 24 hour time interval. In some cases, there was not enough data, i.e. n_{CILM} = 30, but with only 15 days training data available, the training blog postings were cut into n_{CILM} number of segments, while each segment corresponded to *x* number of hours of community interest, where $x < 24$.

The training process compared NDCG10 from top retrieved documents for three groups. The optimized parameter settings were used for the comprehensive re-ranking task.

4.3.3 Re-ranking with baseline and community interest algorithms

The re-ranking task was implemented based on the optimized parameters trained discussed in the last section, meanwhile, the following relevance ranking algorithms were used as a baseline: vector space model, BM25, language model (linear smoothing), language model (Dirichlet smoothing) and language model (two stage smoothing).

Because almost all the relevance based ranking algorithms need to compute the corpus background information, such as IDF or smoothing techniques, in this experiment, a 3GB corpus from a web collection indexed from Google was used. The corpus is a mix of up-to-date web pages and news collected during the evaluation period of time. A list of popular queries (including some testing queries) was used for data collection, and some random retrieved results were indexed (not necessary on the first few result pages) to reduce the bias. The relevance ranking score was computed based on the corpus information. Lemur toolkit was used for this experiment to compute the relevance ranking score for each retrieved document for a given query.

4.4 Evaluation result

4.4.1 Parameter training result

The first task was to train the number of CIV topics and length of time for trend analysis to build the real time interest vector by using NDCG ranking performance. Please note that the number of topics in this research is arbitrarily assigned to each test query because of limited training data, and query specific topic number will be saved for future works, which will be mentioned in Chapter 6. As abovementioned, cross folder validation was used by randomly dividing 45 queries into three groups. The training process repeated for three times. Each time, two groups of queries were used for training, and later the optimized parameter setting was used to test the ranking performance for the third query group. For news search, the result is represented in the following table:

Round 1		day=16	day=18	day=20	day=22	day=24	day=26	day=28	day=30
	t = 30	0. 528205212	0. 530545309	0. 52873765	0. 527733126	0. 531673841	0. 534865606	0. 527929077	0. 530839083
	t = 40	0. 543633681	0. 543697741	0. 546858536	0. 547995524	0. 546241615	0. 545903873	0. 549124882	0. 534850995
	t = 50	0. 531381143	0. 542011477	0. 538148029	0. 540770734	0. 535724017	0. 542340839	0. 541887319	0. 544354378
	t = 60	0. 541314072	0. 542062366	0. 543669953	0. 545023127	0. 544270464	0. 545195867	0. 543488641	0. 545806876
	t = 70	0. 543646143	0. 544995408	0. 541416663	0. 546860907	0. 543318607	0. 546784398	0. 548367999	***0. 550609253***
	t = 80	0. 543576899	0. 544544535	0. 541641336	0. 544796446	0. 54414449	0. 546571395	0. 544761565	0. 546725711
Round 2		**day=16**	**day=18**	**day=20**	**day=22**	**day=24**	**day=26**	**day=28**	**day=30**
	t = 30	0. 530858074	0. 533767696	0. 533158219	0. 52986193	0. 531794693	0. 534624917	0. 527537101	0. 527251676
	t = 40	0. 537205613	0. 540695088	0. 540316766	0. 541118562	0. 541183922	0. 538094572	0. 542440563	0. 536968343
	t = 50	0. 533537171	0. 53474264	0. 53244294	0. 533260322	0. 532174051	0. 529850378	0. 53013393	0. 535423823
	t = 60	0. 536453295	0. 53665808	0. 53387245	0. 536598327	0. 534540396	0. 536971858	0. 535686255	0. 535729852
	t = 70	0. 539919695	0. 539892117	0. 536277659	0. 542273997	0. 536777864	0. 53908274	0. 541269453	***0. 543697084***
	t = 80	0. 540389442	0. 537186595	0. 534505271	0. 539142568	0. 537962097	0. 538197836	0. 538068046	0. 53741228
Round 3		**day=16**	**day=18**	**day=20**	**day=22**	**day=24**	**day=26**	**day=28**	**day=30**
	t = 30	0. 530968336	0. 53288305	0. 531830915	0. 532988894	0. 530681588	0. 530011394	0. 530981475	0. 537460367
	t = 40	0. 54134811	0. 545116296	0. 543113075	0. 542622667	0. 549344178	0. 549572719	0. 546493045	0. 551190925
	t = 50	0. 542241377	0. 538098447	0. 542192074	0. 545962894	0. 543926943	0. 544087018	0. 539839832	0. 547274831
	t = 60	0. 546377591	0. 544185106	0. 546238903	0. 549651079	0. 546121206	0. 549892971	0. 5462577	0. 549893154
	t = 70	0. 538746681	0. 548195724	0. 548162466	0. 546303107	0. 546050761	0. 550609246	0. 554837646	***0. 55533108***
	t = 80	0. 542505187	0. 541597341	0. 542736175	0. 545113674	0. 55087115	0. 544892065	0. 552703625	0. 554670785

Table 4-1. Training CIV for news search

Based on the result, the number of topics = 70 achieved best performance comparing with other number of topics. Meanwhile, $n_{CIV} = 30$ is better than other time, and the ranking performance is increasing averagely when n_{CIV} goes up (except round 2). The best NDCG score is highlighted in the table. To better compare the ranking performance, the result is shown in the following

diagrams (for each training round).

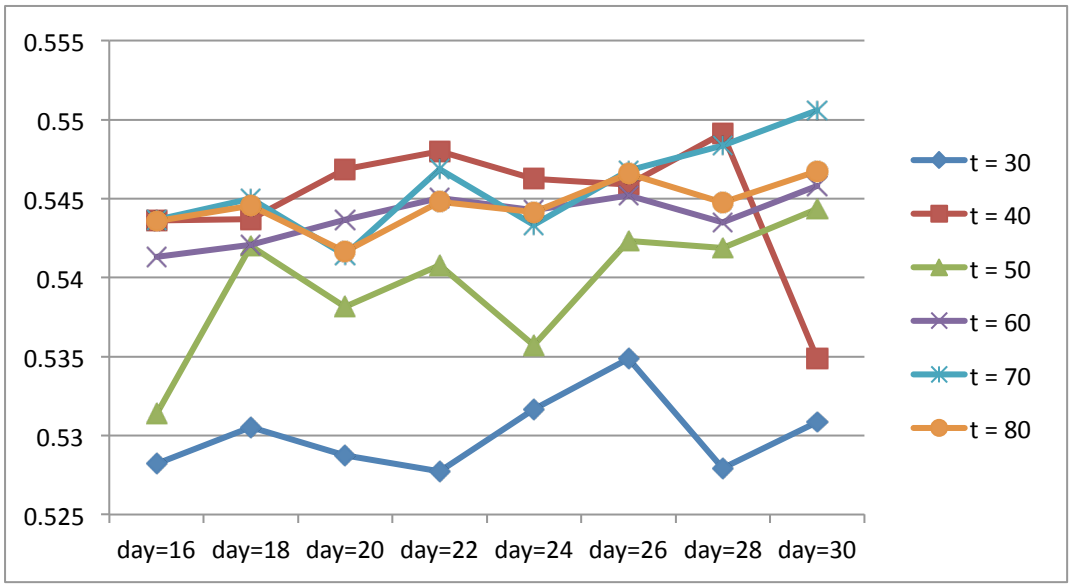

Figure 4-6. NDCG10 for news training (CIV), round 1

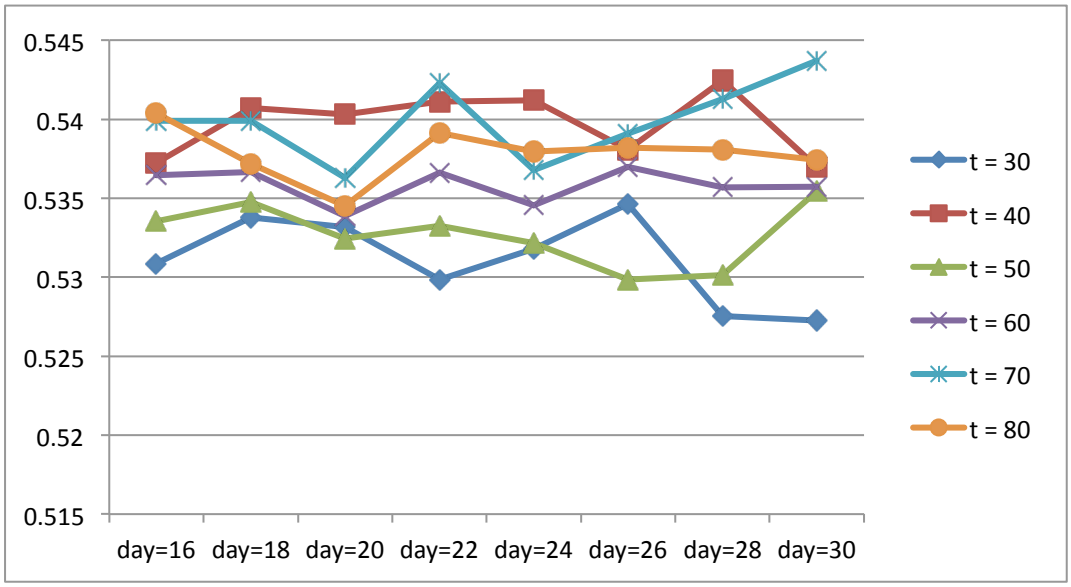

Figure 4-7. NDCG10 for news training (CIV), round 2

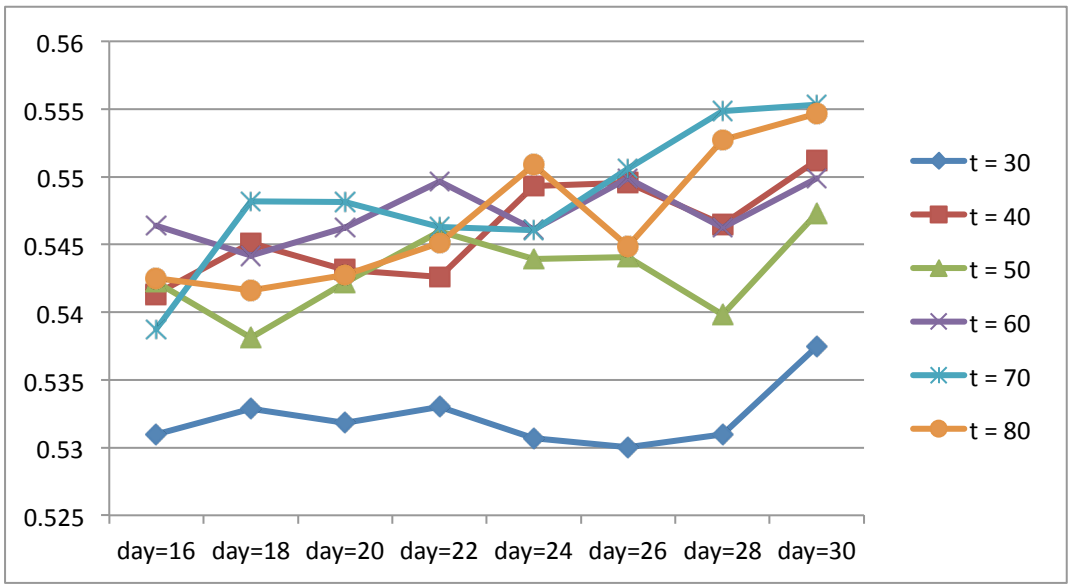

Figure 4-8. NDCG10 for news training (CIV), round 3

Similarly, the web search training result (by using web ranking data) is represented in the following table, and the best performed NDCG scores were highlighted:

Round 1		day=16	day=18	day=20	day=22	day=24	day=26	day=28	day=30
	t = 30	0.432781	0.420947	0.43941	0.420351	0.440045	0.425112	0.420997	0.41781
	t = 40	0.42209	0.429726	0.43476	0.420446	0.432658	0.428395	0.417642	0.41101
	t = 50	0.425773	0.437622	***0.45318***	0.427727	0.412081	0.444762	0.438118	0.42876
	t = 60	0.424103	0.412026	0.42595	0.430342	0.406717	0.426316	0.411313	0.40331
	t = 70	0.403965	0.399901	0.40305	0.404173	0.392796	0.387161	0.389598	0.37806
	t = 80	0.401345	0.399422	0.38867	0.387186	0.387518	0.377709	0.41263	0.40062
Round 2		day=16	day=18	day=20	day=22	day=24	day=26	day=28	day=30
	t = 30	0.485091	0.469388	0.4917	0.49241	0.488796	0.490229	0.480127	0.49374
	t = 40	0.488381	0.50562	0.50873	0.503423	0.493126	0.497168	0.4805	0.48419
	t = 50	0.518337	0.50297	0.52328	0.497814	0.496533	0.515392	0.500564	0.50097
	t = 60	0.500171	0.51364	***0.52954***	0.511185	0.497746	0.51977	0.518056	0.48609
	t = 70	0.493062	0.475814	0.48094	0.487427	0.485164	0.492568	0.485688	0.47061
	t = 80	0.482775	0.477941	0.48387	0.471589	0.477722	0.467142	0.48028	0.48623
Round 3		day=16	day=18	day=20	day=22	day=24	day=26	day=28	day=30
	t = 30	0.499885	0.504103	0.50275	0.485871	0.5234	0.51075	0.500871	0.50362
	t = 40	0.507652	0.50593	0.52311	0.510399	0.503105	0.508757	0.50012	0.50938
	t = 50	0.548039	0.541408	***0.54879***	0.524043	0.526994	0.531059	0.534116	0.53059
	t = 60	0.503093	0.506437	0.52656	0.501632	0.507682	0.506276	0.518057	0.49447
	t = 70	0.498982	0.48029	0.49254	0.498065	0.486658	0.504606	0.506696	0.48629
	t = 80	0.485799	0.487503	0.48199	0.477575	0.490782	0.47363	0.49805	0.48877

Table 4-2. Training CIV for web search

In the web training, we find the number of topics = 50 or 60 with $n_{CIV} = 20$ work best comparing with other parameter setting. Clearly, too large or too small number of topics will threaten the ranking performance. A visualization of the above table is shown in the following diagrams.

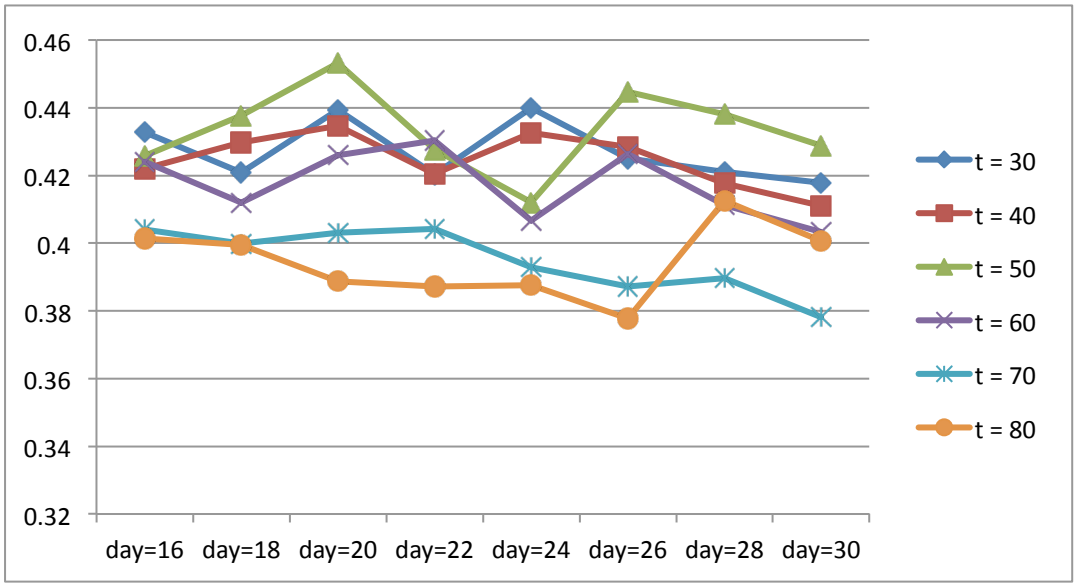

Figure 4-9. NDCG10 for web training (CIV), round 1

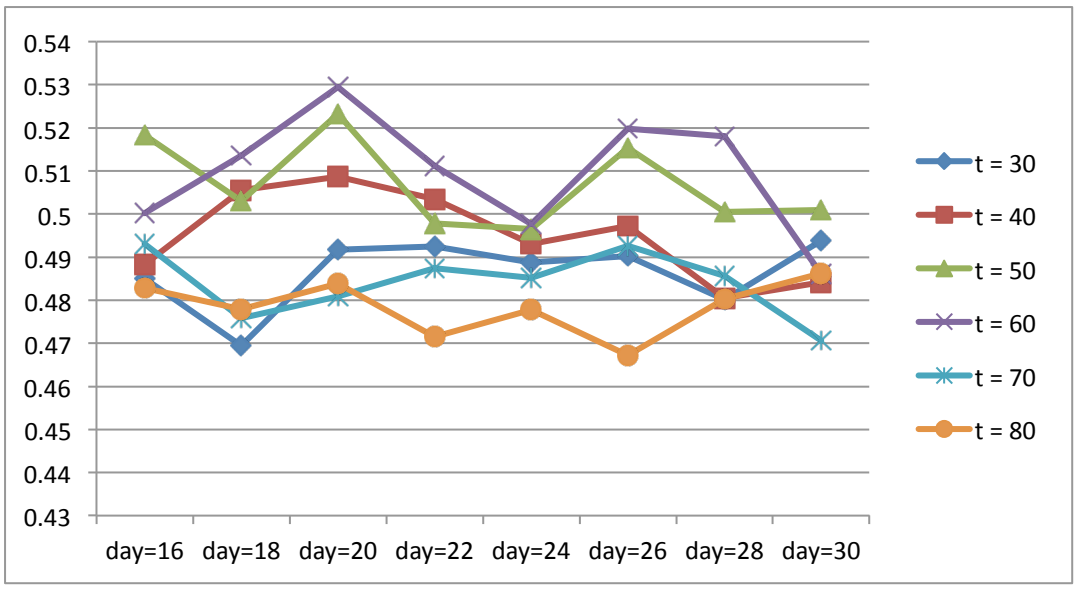

Figure 4-10. NDCG10 for web training (CIV), round 2

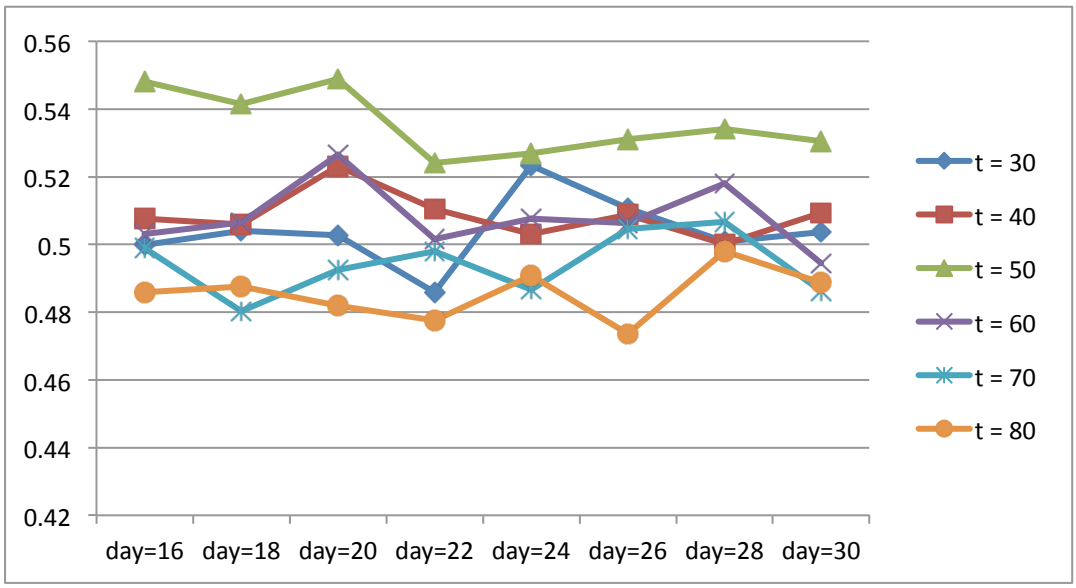

Figure 4-11. NDCG10 for web training (CIV), round 3

Based on the training performance, for web search, the number of topics $t_{CIV} = 50$ or 60 performs best. When the number of topics grows to 70 or 80, the NDCG scores significantly dropped. Regarding the number of days (for trend analysis), for news search, $n_{CIV} = 30$ days empirically works best, while $n_{CIV} = 20$ is optimal for web search. Meanwhile, for news search, we found more training data can, overall, improve the ranking performance. For web search, $n_{CIV} = 20$ works best, and the ranking performance dropped when $n_{CIV} = 30$. However, we didn't find clear trend of change when n_{CIV} changes, as we cannot test $n_{CIV} < 18 \; or \; > 30$. Because CIV algorithm is depend on trend analysis with mean and standard deviation, and too small n_{CIV} will make the result not reliable. Detailed interpretation will be discussed in next chapter.

For CILM, the number of topics (t_{CILM}), the decay parameter (π) and length of time for smoothing (n_{CILM}) was trained. Based on experience from CIV, we first trained the number of topics *t* by using fixed $\pi = 1.0$ and $n_{CILM} = 20$ for query group 1 and 2.

		News	Web
NDCG3	t_{CILM} =30	*0. 349100*	*0. 223699*
	t_{CILM} =40	*0. 346671*	*0. 266725*
	t_{CILM} =50	*0. 366245*	***0. 280963***
	t_{CILM} =60	***0. 369551***	*0. 241847*
	t_{CILM} =70	*0. 366052*	*0. 239529*
	t_{CILM} =80	*0. 357760*	*0. 263981*
NDCG5	t_{CILM} =30	*0. 397763*	*0. 280128*
	t_{CILM} =40	*0. 404028*	*0. 292131*
	t_{CILM} =50	*0. 413060*	***0. 298544***
	t_{CILM} =60	*0. 416877*	*0. 282540*
	t_{CILM} =70	***0. 424131***	*0. 266671*
	t_{CILM} =80	*0. 411443*	*0. 278771*
NDCG10	t_{CILM} =30	*0. 541322*	*0. 359471*
	t_{CILM} =40	*0. 542789*	***0. 379861***
	t_{CILM} =50	*0. 549295*	*0. 379421*
	t_{CILM} =60	***0. 554046***	*0. 358029*
	t_{CILM} =70	*0. 551920*	*0. 341951*
	t_{CILM} =80	*0. 548185*	*0. 364787*

Table 4-3. Training CILM for number of topics t_{CILM}

From the result, we find for news search, the number of topics = 60 or 70 performs best, which the optimized number of topics for web search = 50. The result is similar with CIV training. The result is visualized in the following diagrams:

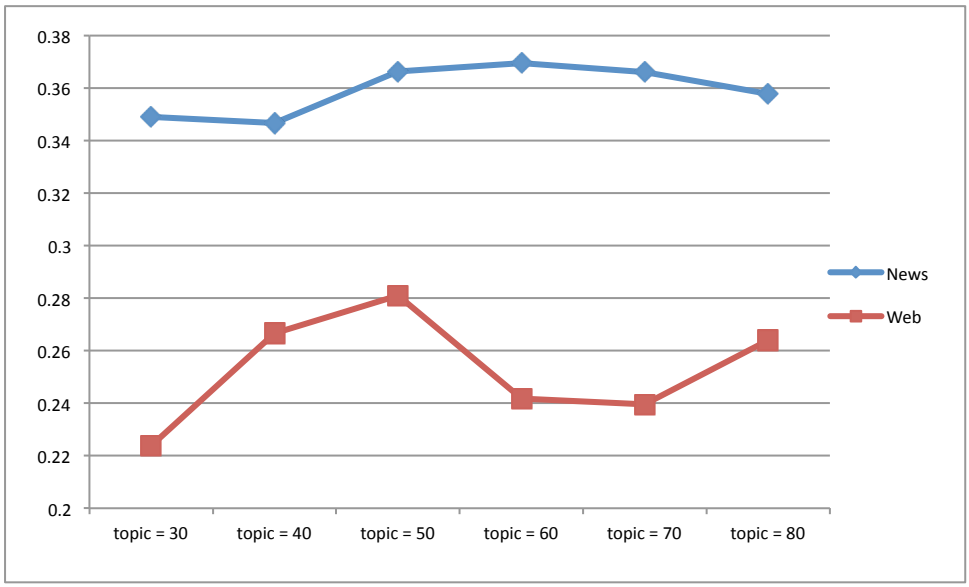

Figure 4-12. NDCG3 for CILM number of topics training

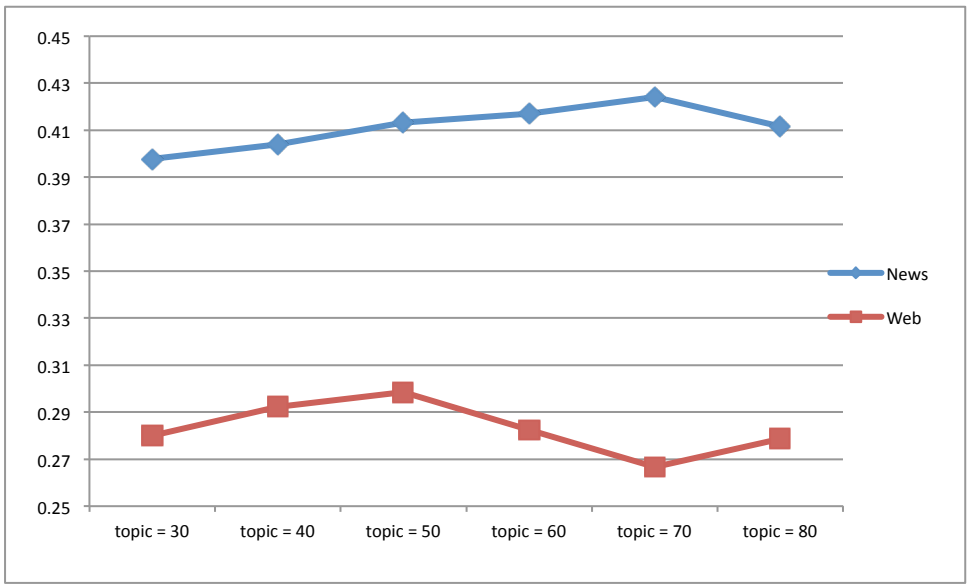

Figure 4-13. NDCG5 for CILM number of topics training

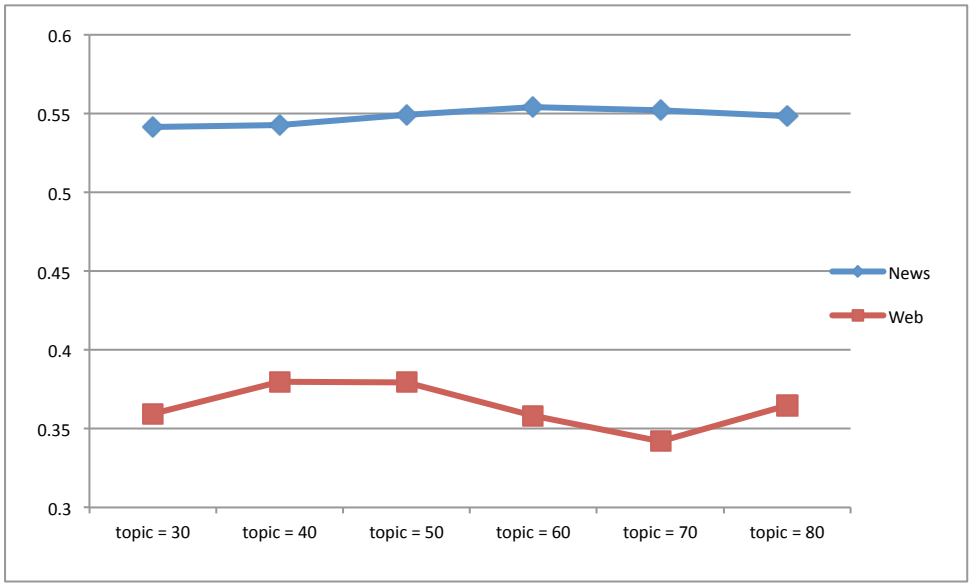

Figure 4-14. NDCG10 for CILM number of topics training

Figures 4-12, 4-13, and 4-14 show that news search is not very sensitive to the number of topics, and $t_{CILM} = 50, 60$ and 70 are slightly better than other numbers of topics (30, 40 and 80). Web search is more sensitive to the change in number of topics. Overall, when $t_{CILM} = 50$, the ranking performs best, which is same as CIV ranking algorithm training method. We may conclude that the number of topic = 50 or 70 can be the optimized parameter to extract semantics of the interest topical space. The possible reason for this will be analyzed in next chapter.

As a result, in the following experiment, we used $t_{CILM} = 70$ for news ranking and $t_{CILM} = 50$

for web ranking as the optimized number to train other parameters and for other evaluation.

With t_{CILM} = *70 or 50*, the length of time for smoothing (n_{CILM}) and decay parameter *(π)* were trained in the following tables. Please note n_{CILM} ranges from 6 to 30, differing from n_{CIV} (from 18 to 30), because for trend analysis, we need more data to compute mean and variation.

For the news ranking:

Round 1		day=6	day=10	day=14	day=18	day=22	day=26	day=30
	π = 0.5	0.53495116	0.540278287	0.544865278	0.537119993	0.547236695	0.541885536	0.545636498
	π = 2.5	0.535634995	0.538641768	0.544170975	0.544293809	0.545094432	0.542862125	***0.55336895***
	π = 4.5	0.531407423	0.539924238	0.544407154	0.548043193	0.544758153	0.54719369	0.551492839
	π = 6.5	0.527693892	0.538461654	0.543251781	0.545417641	0.547679421	0.550084528	0.549541804
	π = 8.5	0.532361512	0.537513328	0.54375911	0.542662566	0.549312178	0.549734659	0.548239541
	π = 10.5	0.525126218	0.536678576	0.539883298	0.542773715	0.546954804	0.549027908	0.551055613
Round 2		day=6	day=10	day=14	day=18	day=22	day=26	day=30
	π = 0.5	0.530225163	0.5306119	0.535280872	0.524948819	0.524881162	0.531055106	0.533813375
	π = 2.5	0.530163142	0.533706275	0.534724353	0.534415837	0.526426851	0.534148551	0.536799926
	π = 4.5	0.525664721	0.536100427	0.539441197	0.540100824	0.534069676	0.536031182	0.53798796
	π = 6.5	0.519276689	0.537436754	0.537311394	0.538589475	0.534594502	0.535514695	***0.540418006***
	π = 8.5	0.521888719	0.531300828	0.53320116	0.536061855	0.534745713	0.535038446	0.538801014
	π = 10.5	0.517126391	0.528845708	0.531876459	0.535094423	0.535138752	0.539877507	0.536905879
Round 3		day=6	day=10	day=14	day=18	day=22	day=26	day=30
	π = 0.5	0.539581064	0.538439219	0.550791611	0.540521535	0.549343588	0.55382362	0.544452161
	π = 2.5	0.542769339	0.542309429	0.548173055	0.543631776	0.545621933	0.552734119	0.539953606
	π = 4.5	0.531848967	0.542744547	0.549014896	0.551493254	0.544460259	0.55071074	0.543533437
	π = 6.5	0.53456453	0.537097229	0.547916821	0.551204001	0.549774011	0.550698033	0.546404581
	π = 8.5	0.533936132	0.53548348	0.545380452	0.550154495	0.551155048	0.55106356	***0.555418275***
	π = 10.5	0.525539375	0.536154753	0.536697819	0.549073898	0.551059549	0.552010639	0.554766886

Table 4-4. Training CILM n_{CILM} and π for news search

Basically, it is a clear trend that the larger n_{CILM} is, namely more historical training data, the better ranking performance will achieve. It is hard to interpret such a large data map, so we first visualize it into the following diagrams, with x-axis representing n_{CILM}.

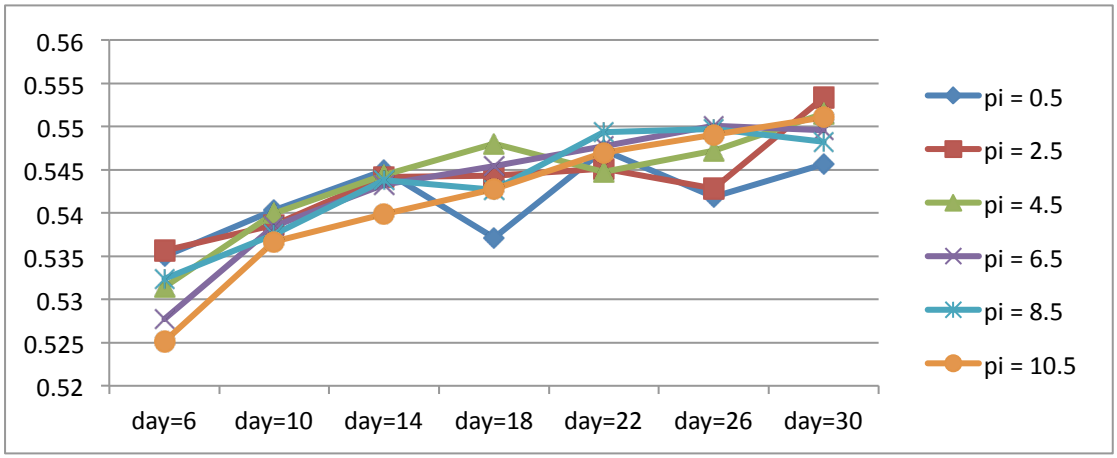

Figure 4-15. NDCG10 for news training (CIV), round 1 (n_{CILM} and π)

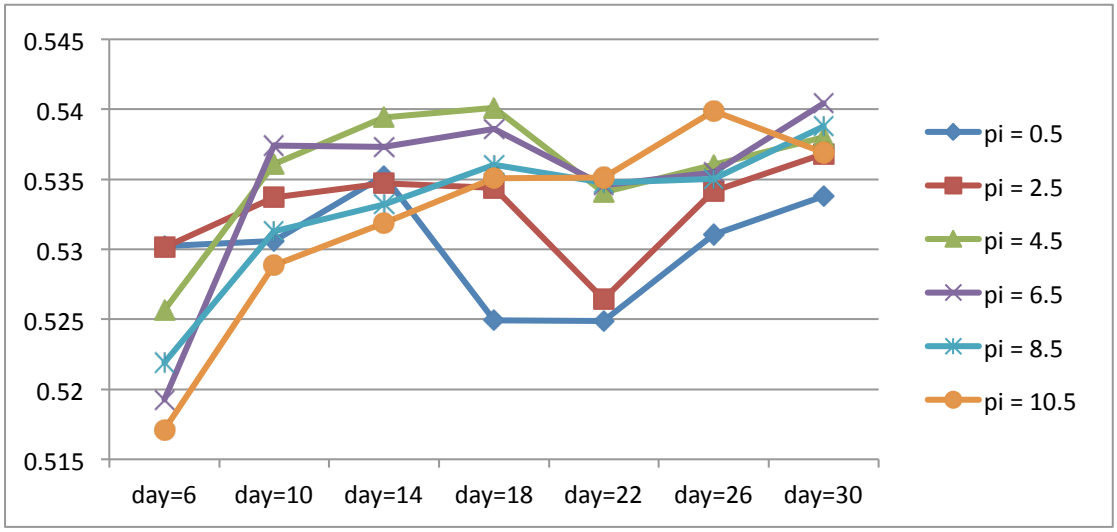

Figure 4-16. NDCG10 for news training (CIV), round 2 (n_{CILM} and π)

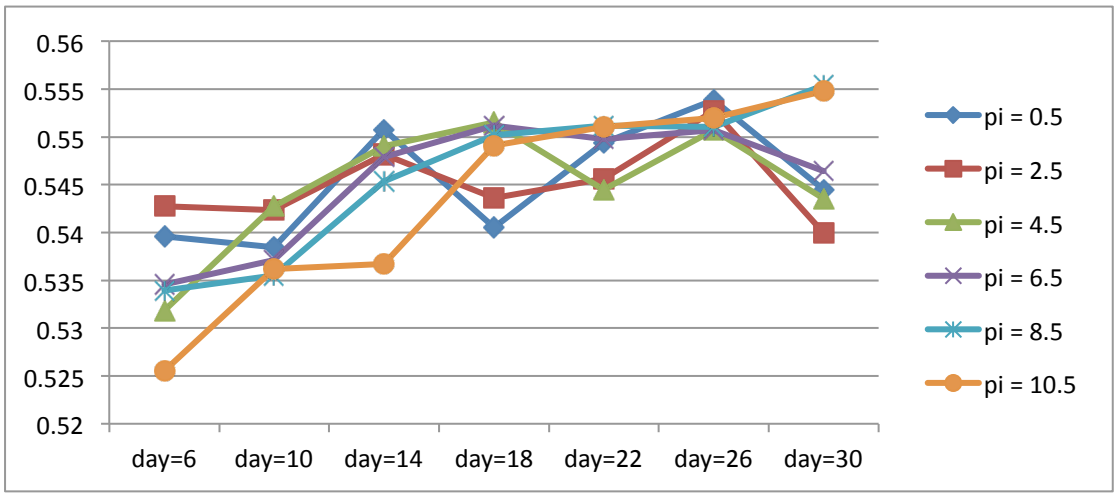

Figure 4-17. NDCG10 for news training (CIV), round 3 (n_{CILM} and π)

Based on the results, we find that the length of time for smoothing (n_{CILM}) is closely related to the

ranking performance. Generally, a longer time for smoothing could improve the news ranking performance, especially for round 1 and 3 training. For the interest decay parameter *(π)*, when π is small (decay speed is fast) the ranking performance is good when n is large (e.g. $\pi = 0.5$ *or* 2.5), but the ranking performance may drop when n decreases. When π is large (decay speed slow), the ranking performance is more stable when $n_{CILM} > 10$, e.g. $\pi = 10.5$ *or* 8.5. Overall, the decay speed training result is not consistent for three training sets. We used the empirically found best performing parameter setting for the next experiment: round 1 *days* n_{CILM} *=30 with* π *=2.5, r*ound 2 *days* n_{CILM} *=30 with* π *=6.5, r*ound 2 *days* n_{CILM} *=30 with* π *=8.5*.

For web search ranking, the results were as follows:

Round 1		day=6	day=10	day=14	day=18	day=22	day=26	day=30
	π = 0.5	0.420975734	0.420717436	***0.422772356***	0.438123029	0.403661447	0.42615354	0.402951477
	π = 2.5	0.408472977	0.41721966	0.407913184	0.413723573	0.419879904	0.423737832	0.397845105
	π = 4.5	0.396747468	0.424784273	0.417278982	0.433162465	0.407985706	0.413598203	0.408817243
	π = 6.5	0.406225346	0.418259059	0.407449943	0.423246204	0.415926863	0.402952781	0.414350631
	π = 8.5	0.408613456	0.432391452	0.422104205	0.418756623	0.411444348	0.409314767	0.407049318
	π = 10.5	0.415918252	0.423009299	0.41693525	0.422414768	0.41681032	0.417895663	0.403618745
Round 2		**day=6**	**day=10**	**day=14**	**day=18**	**day=22**	**day=26**	**day=30**
	π = 0.5	0.489592894	0.481738396	0.497397288	***0.488870937***	0.476453671	0.481739248	0.473006506
	π = 2.5	0.478190406	0.476525526	0.49144487	0.484722468	0.484653402	0.472388027	0.476391197
	π = 4.5	0.464122049	0.485585142	0.486792239	0.482324453	0.480251778	0.482462506	0.485787918
	π = 6.5	0.469138786	0.483461001	0.484240516	0.479767497	0.48028572	0.483279615	0.48193821
	π = 8.5	0.48476698	0.490701965	0.491080635	0.475635253	0.483298219	0.478460869	0.483060105
	π = 10.5	0.485021252	0.480249331	0.487275247	0.485860785	0.485956685	0.488394287	0.484894734
Round 3		**day=6**	**day=10**	**day=14**	**day=18**	**day=22**	**day=26**	**day=30**
	π = 0.5	0.500695757	0.490400187	0.506865688	0.477989341	0.477952689	0.504586805	0.509518585
	π = 2.5	0.495801464	0.494021096	0.505213204	0.492786033	0.483142028	0.491358515	***0.518397812***
	π = 4.5	0.482594249	0.503709968	0.497318464	0.489044715	0.474182874	0.488852375	0.506867045
	π = 6.5	0.480799928	0.496660665	0.498100507	0.484123928	0.47775395	0.476020785	0.499153967
	π = 8.5	0.474790968	0.494147989	0.507664822	0.489629364	0.474808359	0.479087599	0.492598817
	π = 10.5	0.46762839	0.469374624	0.489509126	0.491441156	0.478081881	0.487233194	0.489931175

Table 4-5. Training CILM n_{CILM} and π for web search

Unlike news ranking, web ranking is not very sensitive to the change of n_{CILM}, and larger n_{CILM} does not necessarily lead to better NDCG score. Meanwhile, we find more "randomness" in the

ranking results. The results are visualized in the following diagrams:

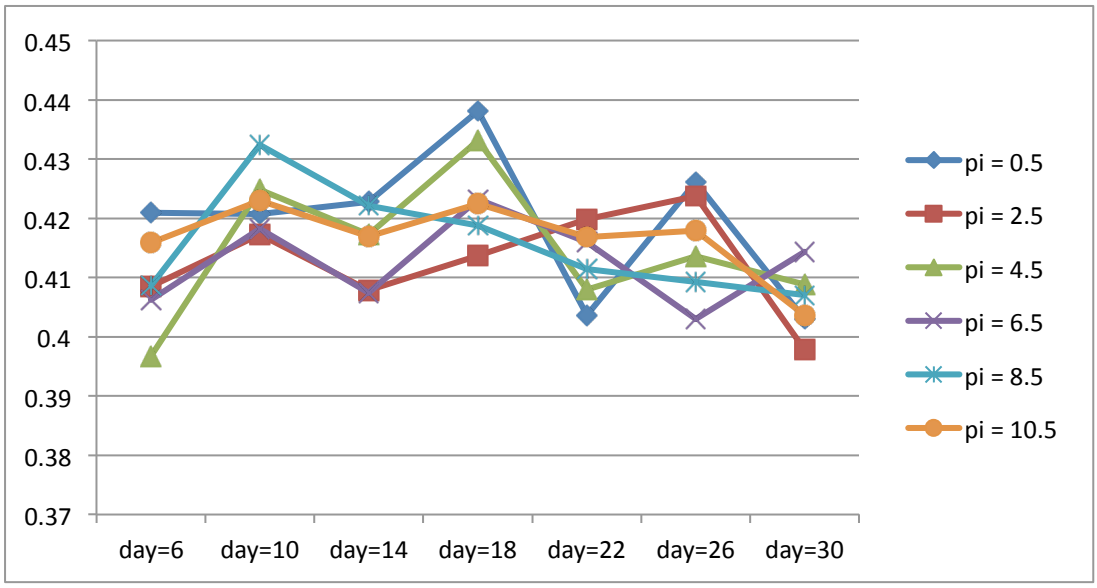

Figure 4-18. NDCG10 for web training (CIV), round 1 (n_{CILM} and π)

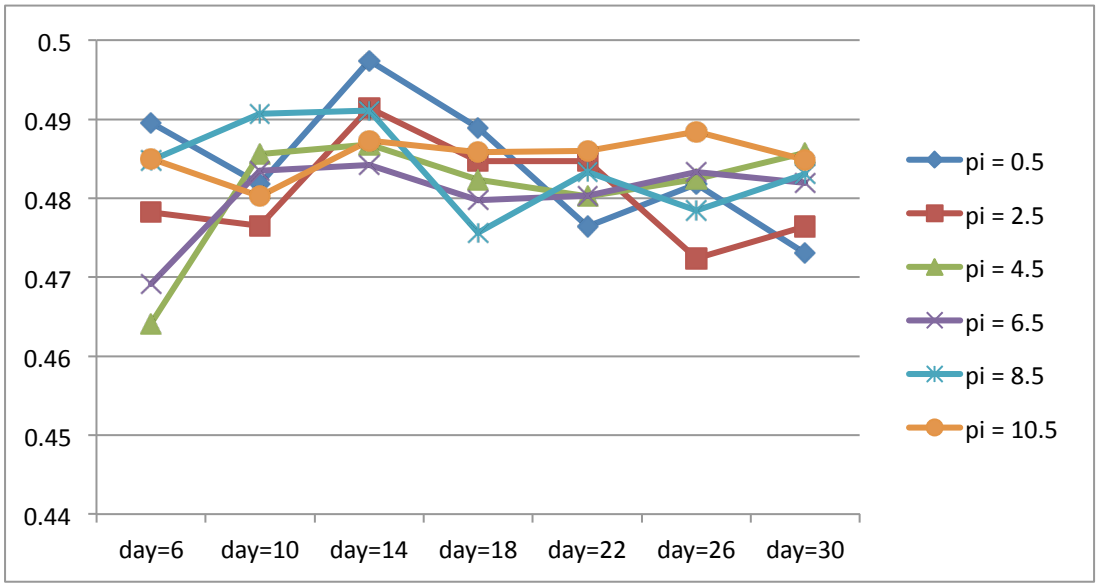

Figure 4-19. NDCG10 for web training (CIV), round 2 (n_{CILM} and π)

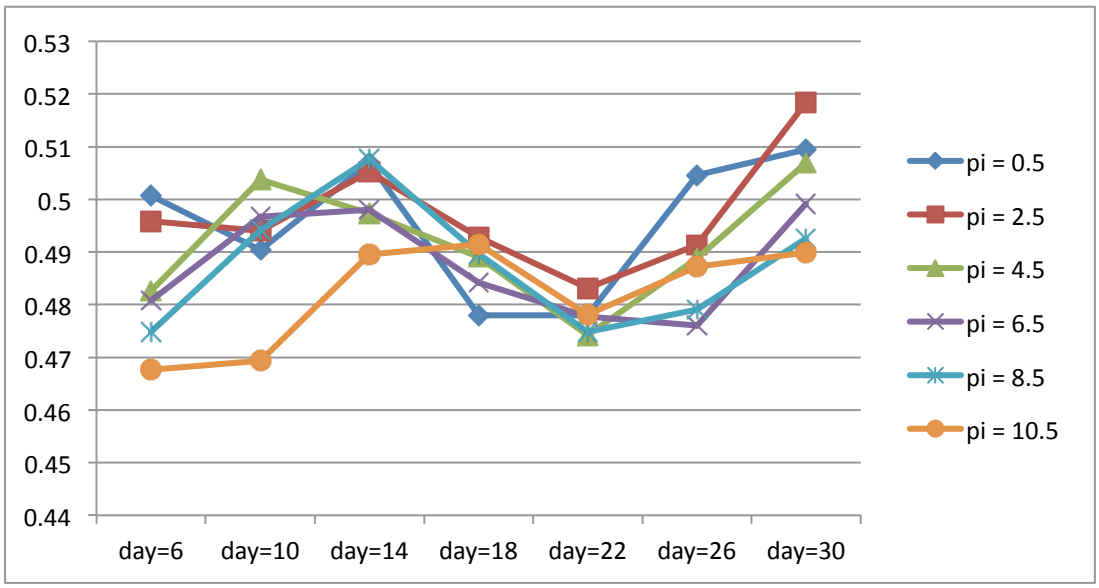

Figure 4-20. NDCG10 for web training (CIV), round 3 (n_{CILM} and π)

For web search, the result looks inconsistent for different parameter settings, especially for decay speed π. But there are still some patterns. First, similar to news search, a larger π makes the ranking performance more stable over differing numbers of days for smoothing, while a smaller π works well for some n_{CILM}. Second, unlike news search, a longer number of days for smoothing doesn't necessary benefit the ranking performance except for round 3. Based on the result, the ranking result peaks when n_{CILM} = *14, 18* and *30* while π =*0.5, 0.5* and *2.5* for these three training folders.

Once again, we employed the empirically best parameter setting in the next experiment. For instance, for round 1, the statistical optimized parameters were trained from query group 1 and 2, and the parameters will be applied to query group 3 for evaluation. The optimized parameter setting for each query group was listed in the following table. The detailed discussion will be analyzed in the next chapter.

Testing	Algorithm	Ranking	Parameter Setting
Query Group 1	CIV	News	t=70, n_{CIV} =30
		Web	t=60, n_{CIV} =20
	CILM	News	t=70, π=6.5, n_{CILM}=30
		Web	t=50, π=0.5, n_{CILM}=18
Query Group 2	CIV	News	t=70, n_{CIV} =30
		Web	t=50, n_{CIV} =20
	CILM	News	t=70, π=8.5, n_{CILM}=30
		Web	t=50, π=2.5, n_{CILM}=30
Query Group 3	CIV	News	t=70, n_{CIV} =30
		Web	t=50, n_{CIV} =20
	CILM	News	t=70, π=2.5, n_{CILM}=30
		Web	t=50, π=0.5, n_{CILM}=14

Table 4-6. Statistical optimized parameter setting for each query testing group

4.4.2 Web Search evaluation

The trained parameters (in Table 4-6) were applied to each query testing group for both the Google web search dataset (45 queries) and the Yahoo web search (45 queries).

The CIV and CILM ranking performance was compared to a list of relevance algorithms' ranking results along with Google or Yahoo ranking results. The results are shown in the following table with t-test results (significance) indicated when the best ranking algorithm is significantly better than other algorithms using the average of NDCG3, NDCG5 and NDCG10).

Google web	NDCG3	NDCG5	NDCG10	t-test
CIV	*0.37168329*	*0.420199822*	*0.500187376*	
CILM	*0.356652652*	*0.387120299*	*0.483420045*	
Google	*0.230423817*	*0.318737414*	*0.388792379*	***
TFIDF	*0.27596245*	*0.333012091*	*0.437831859*	**
BM25	*0.284599431*	*0.336961764*	*0.436466778*	**
LM (liner)	*0.32558799*	*0.382113457*	*0.473992963*	
LM (dirichlet)	*0.34665084*	*0.358128576*	*0.45150825*	
LM (twostage)	*0.349735965*	*0.358725227*	*0.450046444*	*
BEST1:	CIV	CIV	CIV	
BEST2:	CILM	CILM	CILM	
*Significant test *** $t < 0.05$ ** $t < 0.10$ * $t < 0.15$*				

Table 4-7. Ranking performance comparison (Google Web Search)

Yahoo_web	NDCG3	NDCG5	NDCG10	t-test
CIV	*0. 32260902*	*0. 376919597*	*0. 484664526*	
CILM	*0. 391807685*	*0. 40623334*	*0. 492464858*	
Yahoo	*0. 288059321*	*0. 326373542*	*0. 410969176*	**
TFIDF	*0. 24320988*	*0. 282799657*	*0. 404092457*	***
BM25	*0. 245263974*	*0. 277579262*	*0. 395953269*	***
LM (liner)	*0. 276208943*	*0. 316889107*	*0. 432428784*	**
LM (dirichlet)	*0. 223253393*	*0. 270017519*	*0. 385936078*	***
LM (twostage)	*0. 219225991*	*0. 266537146*	*0. 384349848*	***
BEST1:	CILM	CILM	CILM	
BEST2:	CIV	CIV	CIV	
Significant test *** $t < 0.05$ ** $t < 0.10$ * $t < 0.15$				

Table 4-8. Ranking performance comparison (Yahoo Web Search)

For the Google web collection, when relevance based ranking algorithms were applied to the top retrieved documents, the ranking performance improved, especially for language model. CIV algorithm works best for NDCG3, NDCG5, and NDCG10. The averaged CIV is significantly better than BM25 ($t < 0.15$), vector space ($t < 0.15$), Google ranking ($t < 0.10$) and Language Model with twostage smoothing ($t < 0.10$).

For the Yahoo web collection, CILM ranking achieved the best performance followed by CIV. CILM ranking performance is significantly better than vector space, BM25, language model (Dirichlet smoothing), language model (two stage smoothing) at $t < 0.05$, and language model (linear smoothing) at $t < 0.10$.

4.4.3 News Search evaluation

Similar to web search ranking, trained parameters were applied to news search interest modeling. The ranking results of CIV and CILM are compared in the following tables with other ranking algorithms:

Google news	NDCG3	NDCG5	NDCG10	t-test
CIV	*0. 356490637*	*0. 406718238*	*0. 546777902*	***
CILM	*0. 363134762*	*0. 408748121*	*0. 547608992*	***
Google	*0. 414897946*	*0. 453282795*	*0. 574381182*	
TFIDF	*0. 358224206*	*0. 409760281*	*0. 548609575*	*
BM25	*0. 382405788*	*0. 431366158*	*0. 557465133*	
LM (linear)	*0. 327940847*	*0. 384583295*	*0. 532979429*	***
LM (dirichlet)	*0. 398446677*	*0. 451584299*	*0. 568527226*	
LM (twostage)	*0. 399008856*	*0. 453301336*	*0. 568849791*	
BEST1:	Google	Google	Google	
BEST2:	LM (twostage)	LM (twostage)	LM (twostage)	
*Significant test *** $t < 0.05$ ** $t < 0.10$ * $t < 0.15$*				

Table 4-9. Ranking performance comparison (Google News Search)

Yahoo news	NDCG3	NDCG5	NDCG10	t-test
CIV	*0.38799418*	*0.44002231*	*0.567218049*	
CILM	*0.396895839*	*0.441480612*	*0.570672097*	
Yahoo	*0. 357108756*	*0. 408724494*	*0. 553749542*	**
TFIDF	*0. 362768014*	*0. 404059938*	*0. 555076813*	***
BM25	*0. 363704662*	*0. 412726764*	*0. 556132169*	**
LM (linear)	*0. 36258229*	*0. 40508942*	*0. 553391554*	***
LM (dirichlet)	*0. 389559974*	*0. 435801618*	*0. 566983045*	
LM (twostage)	*0. 393606438*	*0. 43828928*	*0. 568684439*	
BEST1:	CILM	CILM	CILM	
BEST2:	LM (twostage)	CIV	LM (twostage)	
*Significant test *** $t < 0.05$ ** $t < 0.10$ * $t < 0.15$*				

Table 4-10. Ranking performance comparison (Yahoo News Search)

Unlike the other groups, Google news result shows that Google news ranking itself is a robust ranking method, performing significantly better than CIV, CILM and language model (linear smoothing) at $t < 0.05$. Language model (two stage smoothing) achieved the second best performance among the candidate ranking algorithms. In this experiment, CILM algorithm ranking performance is better than vector space and language model (linear smoothing), but not significantly. As mentioned earlier, the Google ranking methodology is a black box for us, but we will try to interpret the possible reasons why interest based ranking algorithms fail for news search in the next chapter.

For Yahoo news ranking, CILM achieved the best performance, followed by language model (two stage smoothing). CILM ranking performance is significant better than Yahoo news ranking, vector space, BM25 and language model (linear smoothing), $t < 0.05$ or $t < 0.10$.

A summary of the experimental results from the four groups of data are shown in the following table (a total of 12 time results, each group analyzed at the three different ranking indicators, NDCG3, NDCG5 and NDCG10). The numbers of the two top algorithms are listed. The best performed ranking method was scored as a 3, while the second best ranking algorithm was scored as a 1. The final scores were computed in the last column each line.

	BEST 1	BEST 2	Score
CIV	*3*	*4*	*13*
CILM	*6*	*3*	*21*
Provider (Google or Yahoo)	*3*	*0*	*9*
TFIDF	*0*	*0*	*0*
BM25	*0*	*0*	*0*
LM (linear)	*0*	*0*	*0*
LM (Dirichlet)	*0*	*0*	*0*
LM (twostage)	*0*	*5*	*5*

Table 4-11. Comparison for all ranking methods

The results show that CILM is the best ranking method in this evaluation, while CIV is the second best method. The quality of Google ranking, especially for news, is also good in this test. Among relevance based ranking algorithms, language model (two stage smoothing) works best. However, the results are not completely consistent. Interest based ranking algorithms failed in Google news ranking experiment.

Chapter 5: Discussion and Conclusion

5.1 Introduction

In essence, this thesis is trying to investigate if real time community interest can be modeled (research question 1) and if real-time retrieval ranking can be improved by employing the query specific computational community interest model (research question 2). The evaluation results show positive but not consistent ranking performance across different datasets. In this chapter, we will analyze and draw conclusions about the experiments' results that are presented in Chapter 4.

5.2 Results analysis

In this section, we will analyze the experiments' results. First, interest modeling with parameter training will be investigated to response to research question 1, and then the evaluation results will be interpreted to answer the second and third research questions. At last, we will discuss the evaluation method.

5.2.1 Training parameter setting analysis

The first research question is: *What is community interest? And can we extract and computationally model real time community interest from user textual data?* In this thesis, the community interest is defined as a dynamic distribution of topics over a specific query. Because of the limited size of the training data, we cannot train query specific parameters. For each interest modeling algorithm, CIV or CILM, the optimized parameter setting is trained with news and web search. The performance of the interest modeling will be evaluated by RQ2 and RQ3.

For parameter training, we find the number of topics is related to the ranking performance, as shown, for instance, in Figures 4-12, 4-13, and 4-14. Numbers of topics that are either too large or too small threaten the ranking performance, while number of topics = 60 or 70 leads to the best NDCG scores for news search and number of topics = 50 works best for web ranking. Using the

LDA algorithm, when the number of topics is large, the word distribution, *P(word|topic)*, is dense, and when the number of topics is small, the word distribution is sparse. The training results also show that compared with news search, web search may be more sensitive to the change in the number of topics. Observation suggests that the difference among top ranked web search documents' word distributions could be larger than those of news search (as the news retrieved results are likely to be focusing on particular news events and web retrieved results could come from different domains), so the web search ranking may be more sensitive to an optimized topic model, and the interest probability score may be more sensitive to the performance of the parameter setting of topic modeling.

Another important parameter in interest modeling is the length of time for trend analysis (n_{CIV} for CIV) or historical smoothing (n_{CILM} for CILM). For instance, Figures 4-15, 4-16, and 4-17 show that n_{CILM} is closely related to the ranking performance. The longer the historical data used for smoothing the better is the news ranking performance. Compared with news ranking, web ranking performance (i.e. Figures 4-18, 4-19, and 4-20) is not closely related to the length of time, and the ranking performance is somehow "flat" when n_{CILM} changes. However, from practical perspective, we don't want *n* very large, i.e. more than 30, for a couple reasons. First, community interest ranking is targets currently popular queries, and we cannot predict the popularity of the current query using interest data from long ago. For instance, it would be difficult to predict if a news event will be popular using interest data from before the event occurred. Second, IR system needs to index additional user generated text to train interest model, and we don't want a large amount of historical training data to create too much system load.

Once again, we can interpret this effect with the different characteristics of web and news search. From the language model perspective, the interest score is defined as:

$$\log \frac{P(doc|\theta_{CI-now})}{P(doc|\theta_{CI-history})} \qquad (5\text{-}1)$$

where the smoothing factor is defined by a list of historical interest snapshots, which are trained with blog posting data. As mentioned in sectioned 3.3.3, there are different kinds of topics in the interest model. The ranking function attempts to boost the current "hot topic" while punishing the "diminishing topic" or "background topic" by employing methods like IDF or smoothing. For

news ranking, the distribution of current blog postings (θ_{CI-now}) and news documents are similar, as they both target up-to-date news events given a query. As a result, the more data in the smoothing part, $\theta_{CI-history}$ to reflect the longer history interest distribution (or reflect the longer history semantics), the easier to discriminate the news document that fits the community's current interest. For web search, the web retrieved document distribution may differ from blog postings (θ_{CI-now}) significantly, and the interest scores could be mainly defined by a few key words in the documents, resulting in more "randomness". On the other hand, if we use data from too long a time period for smoothing, i.e. past 27 or 30 days, some "random" words representing more recent interest, but which are still too old for currency in the model, e.g. the past 15 or 12 days, will be chosen to boost the ranking score. This may threaten the ranking performance. As a result, a smaller or reasonable number of n_{CILM} or n_{CIV} may help the interest ranking algorithm to pick some distinctive words, if not the entire interest word collection, which likely represents the most recent community preference. In brief, the news ranking score is mainly decided by the news document distribution (more history data will help), while the web ranking score is decided by some particular words (a reasonable n_{CIV} or n_{CILM} may help). This interpretation will need more experimentation in the future.

The decay parameter π controls the interest decay speed for smoothing in CILM, as shown in Figure 3-11. In this experiment, we find that the larger π will make the ranking performance more stable (over different n_{CILM} settings), but not necessarily make the ranking performance better, as Figures 4-18, 4-19 and 4-20 show. This makes sense given the definition of the decay parameter. When π is small, the smoothing interest decay speed is fast, when π is large, the decay speed is slow. For instance, when π = 0.1, the interest decay speed ≈ 11; when π = 10, the interest decay speed ≈ 1.1. As a result, when π is large all the history interest snapshots make an almost even contribution to $\theta_{CI-history}$. The advantage for a large π is that the $\theta_{CI-history}$ covers all the semantics for the past *n* days evenly, and the smoothing function will identify the important words by contrasting them against all the past interest snapshots. The limitation for a large π is, sometimes a user's interest will last for a while, such as 2 or 3 days. However, given a large π, the current interest (i.e. today) may be diluted by considering too much for the most recent history (i.e. yesterday). A solution is to make π smaller, which

will make longer history more important to $\theta_{CI-history}$ compared with more recent historical snapshots. In the experiment, nevertheless, we did not find significant or consistent improvement by using a smaller π. One possible reason is that differences in the queries should have different decay speeds. However, in this research, we do not have enough training data to decide the query level or query cluster level π_q. This work must be done in the future.

5.2.2 Ranking performance analysis

Regarding the second research question, *in what ways can real-time community interest be used to rank the retrieved results?*, and part of the third research question, *can the community interest based ranking method improve results over existing ranking methods?*, the real-time interest based ranking performance is compared to the rankings of other relevance algorithms and against search engines' rankings.

In the four groups of experiments, the community interest ranking method achieves the best performance in three groups, and the CILM algorithm ranking performance is relative better than CIV in three out of four groups. For web ranking, both Google and Yahoo evaluations show that interest based ranking is an effective method. Meanwhile, in most cases, language model (relevance based) can improve search engines' ranking performance. As mentioned earlier, web ranking is not as dynamic as the news ranking, and the content of top ranked web documents can be different from the training blog postings. In most cases, the web ranking score is defined by key words in the documents but not by the whole distribution.

One remaining question is why community interest does not work well for the Google news search group. Why is the interest ranking better for Yahoo but worse for Google? That suggests we should find out what kinds of queries are good for interest ranking and which are not.

Recall that the 45 test queries were grouped into three sets based on the definition in 4.2.1. For "*World Cup*" queries (11 queries in this group), we target the very short time periods for user interest change (i.e. hourly or several hours of user interest). For "*Recent Popular News*

Event" (15 queries), community interest is extracted based on very recent news events. For "*Long-lasting Popular News Stories*" (19 queries), we target the long lasting popular news stories or news protagonist. The testing queries were presented in Table 5-1

World Cup Queries	World Cup Italy	World Cup Germany
	World Cup Netherlands	World Cup Final
	South Africa	World Cup Argentina
	World Cup Championship	World Cup Brazil
	World Cup Spain	World Cup North Korea
	World Cup	
Recent Popular News Event Queries	Dow Jones	Mel Gibson
	unemployment	stock market
	Lebron	Tiger Woods
	Lady Gaga	wikileaks
	Lindsay Lohan	Nasdaq
	lohan	BP
	Oil spill	NBA Trade
	North Korea	
Long-lasting Popular News Stories Queries	Soccer	Obama
	Car	bullock
	Kobe	california
	wall street	American Idol
	Yahoo	Celtics
	NBA	Golf
	Palin	Economy
	iPhone	China
	Nintendo	Games
	Movie	

Table 5-1. Testing query categories

In the following tables, from the interest based ranking perspective, the number of positive interest rankings (those that are better than Google or Yahoo rankings), and negative instance rankings (those that are worse than Google or Yahoo rankings) percentages are illustrated:

Google News Rank	Positive	Negative	Positive %
World cup	*1*	*10*	*9.09%*
Recent Popular News Event	*9*	*6*	*60.00%*
Long-lasting Popular News Stories	*9*	*10*	*47.37%*

Table 5-2. Positive and negative queries for Google News ranking

Yahoo News Rank	Positive	Negative	Positive %
World cup	*4*	*7*	*36.36%*
Recent Popular News Event	*9*	*6*	*60.00%*
Long-lasting Popular News Stories	*16*	*3*	*84.21%*

Table 5-3. Positive and negative queries for Yahoo News ranking

The above table (Table 5-3) shows the difference between the NDCG10 ranking score for each individual query in three categories, using the difference between the interest based ranking (CILM algorithm) and Google or Yahoo (search provider) ranking ($NDCG10_{CILM} - NDCG10_{provider}$). The score is negative for negative instances, while the score is positive for positive instances.

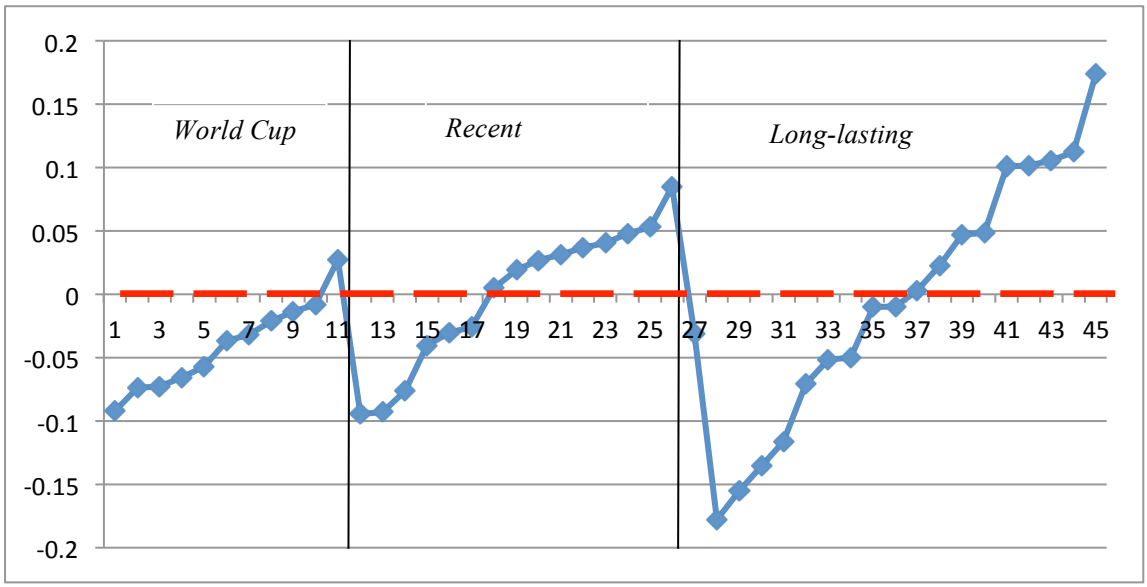

Figure 5-1. Three categories' performance for Google News ranking

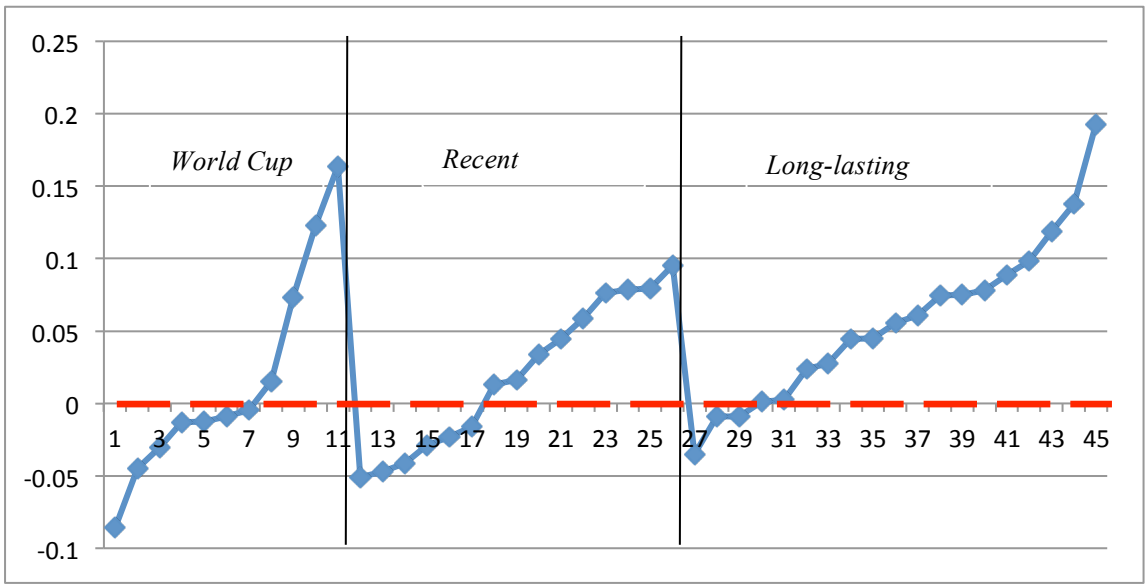

Figure 5-2. Three categories' performance for Yahoo News ranking

For the *World Cup* group, interest ranking performance is poor (only 9.09% positive for Google and 36.36% positive for Yahoo). Interestingly, Google and Yahoo ranking evaluation performance on *Recent Popular News Event* are equal, both achieved 60% positive. The main difference exists in *Long-lasting Popular News Stories* group. For Google, there are only 47.37% positive queries, while for Yahoo there are 84.21% positive queries.

The results can be interpreted as following.

First, unlike our hypothesis, interest modeling is not effective for very short or very dynamic (i.e. hourly) interest changes, such as for the *World Cup* group. A possible reason is that bloggers do not have enough time to update their blog postings to reflect this very dynamic interest change. The evaluation took place at 2:00pm every experimental day, which is the extract time when world cup games began. The community's interest can change dramatically during the game. Meanwhile, the blog search engine may need some time to index those up-to-date blog postings, delaying the interest modeling process, negatively affecting the modeling accuracy. In order to solve this problem, the interest training text data should use a more dynamic kind of real time textual data to train the user interest model, for example, Twitter[6] data. More details will be mentioned in the next chapter.

[6] http://twitter.com/

Second, interest modeling is effective and reliable for recent popular news events as shown in the evaluation results. Some recent popular queries are relatively new for search engines, such like "BP" and "oil spill", which may threaten some existing ranking methodologies, such like clickthrough or user behavior. User interest may or may not change every day. Interest modeling can help search engine better understand communities' interest shift while improving ranking.

Last but not least, Google and Yahoo evaluation results on long-lasting popular news stories group are significantly different. While both search engines' ranking mechanisms are black boxes for us, we can try to analyze why this is the case. One possibility is that Google is the world's largest search engine, and may benefit from the huge amount of users in its ranking function. For instance, a very large amount clickthrough data may help Google ranking achieve a higher NDCG score in the top retrieved documents. Another possible reason is that Google ranking may employ trustworthy news agencies to improve the ranking results. This may be particularly helpful for those long-lasting news stories. However, without data or experiments to support them, these conjectures will remain untested and unverified. We will talk about it in next chapter.

5.2.3 Evaluation method analysis

The remaining research question is the first part of research question 3, *how can we evaluate real-time community interest ranking results?* We can generalize this question as how can we evaluate the real-time ranking method effectively with a low cost?

Obviously, the most accurate way is to use a large number of users for a real-time evaluation focusing on each query and document pair. But the cost of this method is very high. Meanwhile, the current automatic evaluation corpus cannot satisfy the data needs for this dynamic evaluation task. The compromise is to use Amazon Turk for a user evaluation with a reasonable number of queries. The limitation, as mentioned in the last chapter, is that the quality of these turkers and their work cannot be tightly controlled.

In this evaluation, we employed a total of 388 turkers who worked on this task. In most cases,

turkers were able to finish the task within a short amount of time, i.e. 3 to 5 hours, based on requirements. This guaranteed that the interest judgments were up-to-date. But we found these turkers may not be very reliable. A large percentage of turkers only worked on one or two HITs and then left, and we are not sure if some of turkers just provided us "random judgments". In this thesis, we were only able to control for the turkers' location (US turkers), their pre-study abandonment rate and their approval rate. As mentioned earlier, in this research, the average agreement rate of turkers on certain task is 0.548. It proves that a large number of turkers agree with each other when provide their interest judgments.

In the preliminary evaluation, we tested 9 queries (listed in Table 3-5) with 5 high quality judgers (experts). Before the evaluation began, we trained judgers about the evaluation, give them examples of interesting documents and not interesting documents given a query, and give them practice before the real evaluation. For the following 5 days, experts logged in to the evaluation system, and provide their interest judgments given each query and document pair. The details of this experiment can be found at (Liu & von Brzeski, 2009). The following table shows the NDCG3 and NDCG5 performance for Yahoo news ranking and CIV ranking.

	NDCG@3	**NDCG@5**	**Significant test**
Yahoo	0.5740	0.7597	$p < 0.05$ significant
CIV	0.8619	0.8874	$p < 0.1$ significance

Table 5-4. Preliminary evaluation with five experts

The results in Table 5-3 shows, comparing them with the Yahoo news ranking, that CIV can significantly improve the ranking performance (significance analyzed with t-test). This result is similar to the Amazon Turk evaluation over two weeks for 45 queries with a larger number of judgers.

In sum, so far, the evaluation with Amazon Turk for the real-time tasks is effective for the dynamic ranking problem and has a low cost. But we will be looking at other alternatives in the future. Details will be mentioned in the next chapter.

5.3 Conclusion

Community interest ranking can be categorized as a kind of feedback ranking methodology. Unlike traditional relevance feedback ranking methodologies, the extracted real-time community interest is used to provide additional ranking information for the short and popular queries for news or web search engines. Please note that “interest” and “relevance” are related but different for a given query, because a user is interested in a document only if this document is relevant to the query, but interest is a kind of dynamic, user oriented information that comes from external textual data (not retrieved documents). An advantage for this ranking method is we do not need to ask users to provide interest judgments toward each query, which makes the ranking cost low. We can refer to this method as “pseudo interest feedback”. Figure 5-3 shows the differences among “pseudo interest feedback”, user feedback and pseudo relevance feedback. First, pseudo relevance feedback gets the ranking information from retrieved results, while interest feedback gets the ranking information from outside textual resource. Second, user feedback needs direct user intervention and thus has a high cost, while interest feedback uses user generated real-time texts to represent the user, that is to say interest feedback uses “pseudo users”.

Another advantage of the interest ranking model is that the computational community interest model is extracted from chronological user generated textual data, such as blog postings, data that is available without usage boundaries. This is very important for academic research and small search engines without access to the large amount of user data. Last, this is a dynamic feedback ranking method, as community interest changes over time.

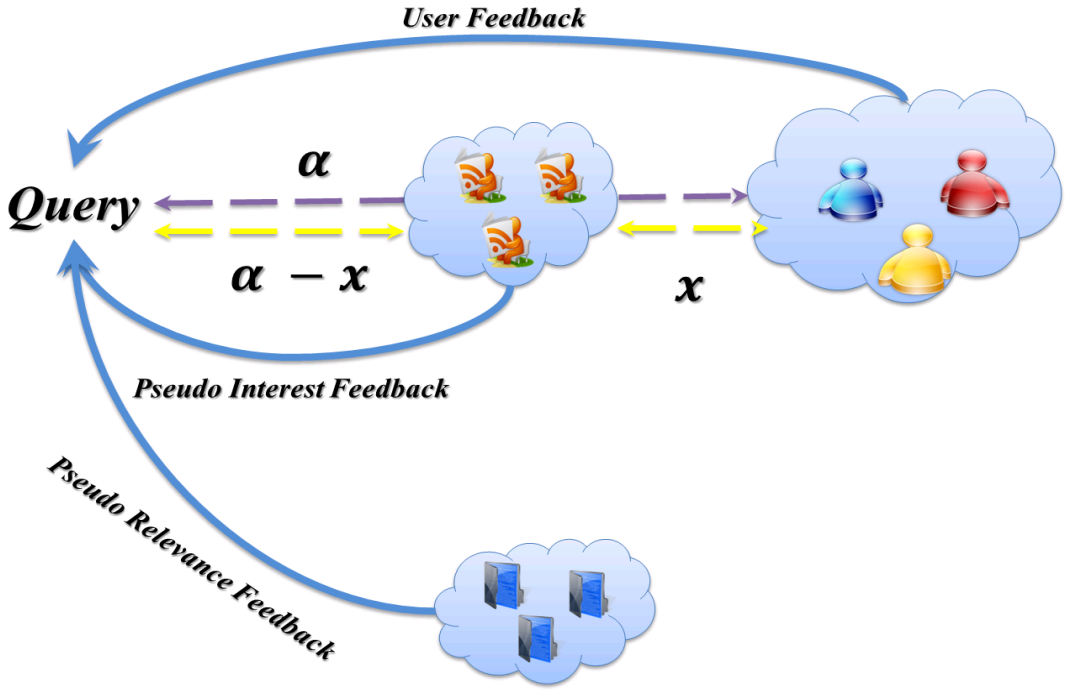

Figure 5-4. Compare three different feedbacks

As mentioned in the first chapter, a great challenge of IR is to figure out the distance between a user's information need and a query (distance α). As Figure 5-4 shows, if the user generated chronological textual data can somehow represent a user's information need (i.e. distance = x), the original distance α will be shortened. A better interest ranking algorithm or a higher quality training data will result in a smaller x while improving the ranking performance.

In this thesis, user interest is defined as a dynamic distribution or vector over topic space, while the topic is defined as probability distribution over words. The advantage for topic modeling is that it can help us to identify the topic(s) of the new words. This is especially helpful for news ranking. If a news or web document concentrates on topic z, for instance, but the page includes a list of new words that never appear in the interest training data, traditional ranking algorithms can have difficulty computing the ranking score effectively. By using topic modeling algorithm, we can infer the probability of z given a document by using other words in the document. However, in the experiments, we also find that topic modeling itself is not perfect. The algorithm complexity and challenges in parameter setting may result in unstable topic model precision, harming the ranking performance. On the other hand, topic modeling algorithms, like LDA, need a large amount of training data, which may not be available for some queries.

As mentioned earlier in this chapter, it is an arbitrary decision to use unique parameter settings for different queries for news or web ranking. The evaluation shows that different queries have different natures, and ranking performance is likely to be improved by using different parameter settings, such as different numbers of topics and different smoothing factors. However, the size of the training data does not support training query level or category level parameters. This should be saved for future work.

The evaluation results show that, overall, interest based ranking is an effective ranking method to deal with both news and web ranking problems. Based on NDCG evaluation, interest ranking is statistically better than other ranking methods (relevance ranking and search engine ranking), except for the Google news group.

Chapter 6: Future work

6.1 Introduction

As mentioned above, while this thesis can make both theoretical and practical contribution to information retrieval ranking research, there are also some limitations. In this chapter, we will propose several possible directions for future research.

6.2 Future directions

6.2.1 Query level interest parameter setting

First, in these experiments, we found the characteristics of user's interest toward different queries may be relatively either dynamic or stable, which results in different parameter settings for different queries or different query categories. We could train query level or query category level parameters by implementing a more comprehensive evaluation.

In the error analysis part, i.e. Table 5-1 and 5-2, we found different kind of queries may have different interest characters. For instance, for a sport related query, users' interest may change quickly when the game score is changing. On the contrary, a political query's user interest may be more stable. In order to better mirror the real world community interest change toward different queries, we need to train different query level parameter such like query interest decay speed.

With more training data and specialized parameter setting, the ranking performance could be further improved. However, user involved parameter training process is still too expensive. We may need a more automatic method to train the interest parameters. For example, the speed of topic change in social media toward the target query could be used as an indicator to reflect the interest decay speed. This work should be saved for the future research.

6.2.2 Interest training data

We used dynamic blog postings for community interest training in this thesis. However, we found, sometimes a blog is not good enough to mirror very dynamic user interest change, for instance, the World Cup related queries in news search, where community interest may change every hour

during the game, and users didn't have enough time to update their blogs. As a result, more dynamic user generated textual data could be used to train dynamic community change, i.e., Twitter and Facebook[7] data.

However, employing other kind of training data also has to face some new challenges. Take Twitter as an example, some typical characters of blog and Twitter data are compared in the following table.

	Blog	**Twitter**
Interest Representation	Daily interest or weekly interest	Hourly interest
Semantics	Rich semantic, good for topic training	Short message, can hardly extract topics given existing algorithms
Noise	Close to news distribution	Noisy features…

Table 6-1. Comparison of Blog and Twitter

The advantage of using Twitter data is we can extract very dynamic user interest (like hourly interest) change toward the target query. However, there are two major limitations. First, unlike blog postings, Tweets are very short messages, which are limited to 150 characters. We can hardly use existing topic modeling algorithm to effectively extract interest topic space. Meanwhile, Twitter is more noisy compared to blog. For instance, Twitter users uses informal language to communicate and express their opinion. The extracted interest model (as a distribution over topics or words) could be difficult to directly apply to the ranking function for the news or web retrieved results, as the word distribution of news and web page is different from the interest model. If this is the matter, we may have to add a middle layer between Twitter based interest model and ranking module to translate the word distribution.

6.2.3 Community based ranking

The "community" needs to be better studied in the future.

In this research, global community interest is used for ranking experiments. However, we know different community may have different kinds of interest over the target query. For instance, we

[7] http://facebook.com

can classify the retrieval system users into different virtual communities based on their gender, location and profession. Different community should have different interest model alone with communitized parameter setting when ranking. As a result the ranking function could be:

$$Ranking\ score = P_{interest}(doc|query, community, time) \quad (6\text{-}1)$$

In 6-1, the ranking score is represented by the probability that user is interested in a retrieved document given the query, time and community. In order to achieve this ranking function, we need to know two things: first, how to model community based interest, and how to decide if the system user belongs to a specific community? The prior problem could be solved if we can access user blogger information in this research. For instance, accessing IP address of blogger could help us to generate community interest model for a city or for a country. The later question is hard to be solved if evaluation is not launched by search provider. For a small scale evaluation, we could find a group of users and divide them into different communities by using a simple survey. Later, we could evaluate if applying communitized interest model will help system improve ranking performance. But the cost of this evaluation will be higher than using Amazon Turk.

6.2.4 Automatic evaluation

Last but not least, a more automatic evaluation method can be used to judge the interest based ranking performance. In this research, we used Amazon Turk to judge 45 test queries on different days. It is always desirable to employ fully automatic evaluation processes as TREC does to evaluate a ranking methodology.

The biggest challenge for evaluation is we need real-time user interest (or relevance) judgments, because we assume that users' interest may change over time and the optimized ranking should change as well.

However, we have other opportunities to indirectly evaluate real-time interest based ranking algorithm. For instance, news ranking data could be used to test the algorithm performance. Take New York Times data and "page one rank" problem as an example. Recently, New York Times released their annotated text corpus (Sandhaus, 2008) with print page information (e.g. a piece of news text printed on the first or second page on a specific date). In most cases, an edit team work

on this problem to decide if a specific news can be printed on the first page based on its importance. This "page one rank" problem provides interest based ranking algorithm opportunity for automatic evaluation. As the figure 6-1 shows:

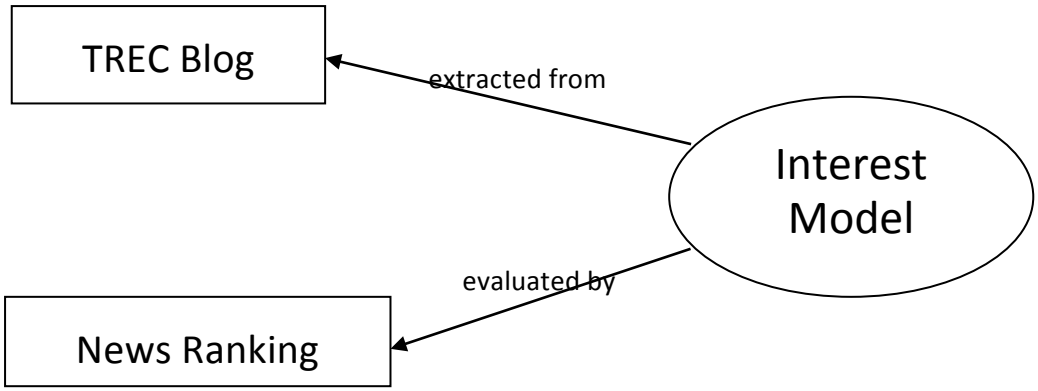

Figure 6-1. Automatic evaluation with News Ranking corpus

We could use TREC blog data as interest model training corpus, and the extracted interest model can be used to predict news ranking problem or to say "page one rank" problem. If interest model is a robust and effective ranking model, it is possible that it can automatically predict the news ranking information.

This evaluation remains two challenges. First, in this thesis, interest model is used for information retrieval problem, and query level interest model is used for tanking. As a result, if we use "page one rank" to evaluate this algorithm, a list of "pseudo queries" is necessary to build the interest models. For example, news tags can be used as a kind of pseudo query. Second, some baseline algorithms need to be chosen to judge the performance of the interest modeling ranking methods.

Reference

Adamic, L. A., & Glance, N. (2005). The political blogosphere and the 2004 U.S. election: divided they blog. Proceedings of the 3rd international workshop on Link discovery.

Adar, E., Zhang, L., Adamic, L., & Lukose, R. (2004). Implicit structure and the dynamics of blogspace. WWW 2005 Workshop on the Weblogging Ecosystem.

Agichtein, E., Brill, E., & Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval.

Alonso, O., Rose, D. E., & Stewart, B. (2008). Crowdsourcing for relevance evaluation. ACM SIGIR Forum 42(2)

Amazon Mechanical Turk, https://www.mturk.com/.

Anh, V. N., & Moffat, A. (2002). Improved retrieval effectiveness through impact transformation. Proceedings of the 13th Australasian database conference.

Aronson, A., & Rindflesch, T. (1997). Query expansion using the UMLS Metathesaurus. Proceedings of the American Medical Informatics Association (AMIA) Symposium.

Azzopardi, L., Girolami, M., & Rijsbergen, K. v. (2004). Topic based language models for ad hoc information retrieval. Proceedings of the 2004 IEEE International Joint Conference on Neural Networks.

Baeza-Yates, R., Hurtado, C., & Mendoza, M. (2004a). Query clustering for boosting web page ranking. Lecture Notes in Computer Science, 164-175.

Baeza-Yates, R., Hurtado, C., & Mendoza, M. (2004b). Query recommendation using query logs in search engines. Current Trends in Database Technology - EDBT 2004 Workshops, 588-596.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). Modern information retrieval. ACM Press, New York.

Bartell, B., Cottrell, G., & Belew, R. (1995). Proceedings of the Swedish Conference on Connectionism.

Beeferman, D., & Berger, A. (2000). Agglomerative clustering of a search engine query log. Proceedings of the 6th ACM SIGKDD international conference.

Beitzel, S. M., Jensen, E. C., Chowdhury, A., Grossman, D., & Frieder, O. (2004). Hourly analysis of a very large topically categorized web query log. Proceedings of the 27th Annual International ACM SIGIR conference on Research and development in information retrieval.

Belkin, N., Oddy, R., & Brooks, H. (1982). ASK for information retrieval: Part I. Background and theory. Journal of documentation, 38(2), 61-71.

Belkin, N., Cool, C., Jeng, J., Keller, A., Kelly, D. Kim, J., Lee, H., Tang, M. & Yuan, X. (2002) Rutgers' TREC 2001 Interactive Track Experie nce. In E.M. Voorhees & D.M. Harman (Eds.) The tenth text retrieval conference, TREC 2001 (pp.465-472).

Belkin, N., Kelly, D., Kim, G., Kim, J., Lee, H., Muresan, G., Tang, M., Yuan, X. & Cool, C. (2003). Query length in interactive information retrieval. Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval.

Blei, D., Ng, A., & Jordan, M. (2003). Latent dirichlet allocation. The Journal of Machine Learning Research, 3, 993-1022.

Blei, D. M., Griffiths, T. L., Jordan, M. I., & Tenenbaum, J. B. (2004). Hierarchical topic models and the nested Chinese restaurant process. Advances in neural information processing systems, 16, 106.

Bookstein, A. (1974). The anomalous behaviour of precision in the Swets model, and its resolution. Journal of documentation, 30(4), 374-380.

Breese, J., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. Proceedings of the Conference on Uncertainty in Artificial Intelligence.

Brown, P., Pietra, V., deSouza, P., Lai, J., & Mercer, R. (1992). Class-based n-gram models of natural language. Computational linguistics, 18(4), 467-479.

Buckland, M., & Gey, F. (1999). The relationship between Recall and Precision. Journal of the American Society for Information Science, 45(1), 12.

Buckley, C., & Voorhees, E. M. (2000). Evaluating evaluation measure stability. Proceedings of the 23th Annual International ACM SIGIR Conference on Information Retrieval.

Burges, C., Ragno, R., & Le, Q. (2007). Learning to rank with nonsmooth cost functions. Advances in neural information processing systems, 19, 193.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N. & Hullender, G. (2005). Learning to rank using gradient descent. Proceedings of the 22nd international conference on Machine learning.

Cao, Y., Xu, J., Liu, T., Li, H., Huang, Y., & Hon, H. (2006). Adapting ranking SVM to document retrieval. Proceedings of the 29th Annual International ACM SIGIR Conference on Information Retrieval.

Carmel, D., Yom-Tov, E., Darlow, A. & Pelleg, D. (2006) What Makes a Query Difficult?, Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval.

Chakrabarti, S. (1999). Recent results in automatic Web resource discovery. ACM Computing Surveys (CSUR), 31(4es), 17.

Cleverdon, C. W. (1972). On the Inverse Relationship of Recall and Precision. Journal of documentation, 28(3), 195-201.

Croft, B. (1993). Knowledge-based and statistical approaches to text retrieval. IEEE Expert, 8(2), 8-12.

Croft, W., Cronen-Townsend, S., & Larvrenko, V. (2001). Relevance feedback and personalization: A language modeling perspective. DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries.

Cronen-Townsend, S., Zhou, Y., & Croft, W. (2002). Predicting query performance. Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval.

Dakka, W., Gravano, L., & Ipeirotis, P. G. (2008). Answering general time sensitive queries. Proceedings of the 17th ACM conference on Information and

knowledge management.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6), 391-407.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society. Series B (Methodological), 39(1), 1-38.

Dumais, S., Letsche, T., Littman, M., & Landauer, T. (1997). Automatic cross-language retrieval using latent semantic indexing. AAAI Spring Symposuim on Cross-Language Text and Speech Retrieval, 115–132.

Evans, B. M., & Chi, E. H. (2008). Towards a model of understanding social search. Proceedings of the ACM conference on Computer supported cooperative work.

Evans, D., Huettner, A., Tong, X., Jansen, P., & Bennett, J. (1999). Effectiveness of clustering in ad-hoc retrieval. NIST SPECIAL PUBLICATION SP, 143-148.

Fellbaum, C. (1998). WordNet: An electronic lexical database: MIT press Cambridge, MA.

Foltz, P. (1990). Using latent semantic indexing for information filtering. Proceedings of the ACM SIGOIS and IEEE CS TC-OA conference on Office information systems, 11(2-3), 47.

Fonseca, B. M., Golgher, P. B., Moura, E. S. d., & Ziviani, N. (2003). Using association rules to discover search engines related queries. Proceedings of the 1st Latin American Web Congress.

Fox, S., Karnawat, K., Mydland, M., Dumais, S., & White, T. (2005). Evaluating implicit measures to improve web search. ACM Transactions on Information Systems (TOIS), 23(2), 147.

Fujimura, K., Inoue, T., & Sugisaki, M. (2005). The eigenrumor algorithm for ranking blogs. WWW Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics.

Furukawa, T., Matsuzawa, T., Matsuo, Y., Uchiyama, K., & Takeda, M. (2006). Analysis

of User's Relation and Reading Activity in Weblogs. Lecture Notes in Computer Science, 4012, 280.

Gordon, M. D., & Kochen, M. (1989). Recall-precision trade-off: a derivation. Journal of the American Society for Information Science, 40, 145-151.

Heine, M. H. (1973). The inverse relationship of precision and recall in terms of the Swets' model'. Journal of documentation, 29, 81-84.

Hersh, W., Price, S., & Donohoe, L. (2000). Assessing thesaurus-based query expansion using the UMLS Metathesaurus. Proceedings of the American Medical Informatics Association (AMIA) Symposium.

Hiemstra, D. (1998). A linguistically motivated probabilistic model of information retrieval. Lecture Notes in Computer Science, 569-584.

Hiemstra, D. (2001). Using language models for information retrieval. University of Twente.

Hofmann, T. (1999). Probabilistic latent semantic indexing. Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval.

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems (TOIS), 20(4), 446.

Jacquemin, C. (1999). Syntagmatic and paradigmatic representations of term variation. Proceedings of theannual meeting of the Association for Computational Linguistics on Computational Linguistics

Jansen, B. J., Spink, A., Bateman, J., & Saracevic, T. (1998). Real life information retrieval: a study of user queries on the Web, ACM SIGIR Forum: ACM New York, NY, USA.

Jardine, N., & van Rijsbergen, C. J. (1971). The use of hierarchic clustering in information retrieval. The use of hierarchic clustering in information retrieval, 7(5), 217-240.

Joachims, T. (2003). Evaluating retrieval performance using clickthrough data. Text Mining. 79-96.

Joachims, T. (2002). Optimizing search engines using clickthrough data. Proceedings

of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. Journal of documentation, 28(1), 11-21.

Jones, K. S. (1973). Collection Properties Influencing Automatic Term Classification Performance. Information Storage and Retrieval, 9(9), 499-513.

Jones, K. S., & van Rijsbergen, C. J. (1976). Information retrieval test collections. Journal of Documentation, 32(1), 59-75.

Karlgren, J. & Franzén, K. (1997) Verbosity and interface

design. Retrieved on 17 January 2002 at:

http://www.ling.su.se/staff/franzen/irinterface.html

Kim, H.R., & Chan, P. K. (2008). Learning implicit user interest hierarchy for context in personalization. Applied Intelligence, 28(2), 153-166.

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM), 46(5), 604-632.

Kraft, R., Chang, C. C., Maghoul, F., & Kumar, R. (2006). Searching with Context. Proceedings of the 15th International World Wide Web Conference.

Kwok, K. (1996). A new method of weighting query terms for ad-hoc retrieval. Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval

Lafferty, J., & Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval.

Lau, T., & Horvitz, E. (1999). Patterns of Search: Analyzing and Modeling Web Query Refinement. International Conference on User modeling.

Lewis, D. D. (1992). An evaluation of phrasal and clustered representations on a text categorization task. Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval.

Li, H. (2002). Word clustering and disambiguation based on co-occurrence data.

Natural Language Engineering, 8(01), 25-42.

Lin, D. (1998). An information-theoretic definition of similarity. Proceedings of the 15th International Conference on Machine Learning.

Liu, T. (2009). Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 3(3), 225-331.

Liu, X., & Brzeski, V. v. (2009). Computational community interest for ranking. Proceeding of the 18th ACM Conference on Information and Knowledge Management.

Liu, X., & Croft, W. B. (2004). Cluster-based retrieval using language models. Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval.

Liu, X., Qin, J., Chen, M., & Park, J.-H. (2008). Automatic semantic mapping between query terms and controlled vocabulary through using WordNet and Wikipedia. Proceedings of the American Society for Information Science and Technology, 45(1).

Liu, Y., Bian, J., & Agichtein, E. (2008). Predicting information seeker satisfaction in community question answering. Proceedings of the 31th annual international ACM SIGIR conference on Research and development in information retrieval.

Luhn, H. (1957). A statistical approach to mechanized encoding and searching of literary information. IBM Journal of research and development, 1(4), 309-317.

Macdonald, C., Ounis, I., & Soboroff, I. (2008). Overview of the TREC-2007 Blog Track. In Proceedings of TREC 2007.

Maglio, P. P., & Barrett, R. (1997). How to build modeling agents to support web searchers. Courses and Lectures-International Centre for Mechanical Sciences, 5-16.

Marchionini, G. (1997). Information seeking in electronic environments. Cambridge University Press.

Metzler, D., & Croft, W. (2004). Combining the language model and inference

network approaches to retrieval. Information processing & management, 40(5), 735-750.

Mishne, G., & de Rijke, M. (2006). A study of blog search. Lecture Notes in Computer Science, 3936, 289.

Mooers, C. (1952). Information retrieval viewed as temporal signaling. Proceedings of the International Congress of Mathematicians.

Ng, K. (2000). A maximum likelihood ratio information retrieval model. Proceedings of the 8th Text REtrieval Conference (TREC- 8), 483–492.

Ounis, I., de Rijke, M., Macdonald, C., Mishne, G. & and Soboroff, I. (2007). Overviewof the TREC 2006 Blog Track. In Proceedings of TREC 2006.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1998). The PageRank Citation Ranking: Bringing Order to the Web. Stanford InfoLab.

Ponte, J. M., & Croft, W. B. (1998). A language modeling approach to information retrieval. Proceedings of the 21st annual international Annual ACM Conference on Research and Development in Information Retrieval.

Praks, P., Dvorsky, J., & Snášel, V. (2003). Latent semantic indexing for image retrieval systems. SIAM Linear Algebra Proceedings.

Qiu, F., & Cho, J. (2001). Automatic identification of user interest for personalized search. Proceedings of the 15th international conference on World Wide Web.

Qiu, Y., & Frei, H.P. (1993). Concept based query expansion. Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval.

Robertson, S. (1997). Overview of the okapi projects. Journal of documentation, 53(1), 3-7.

Robertson, S. E. (1975). Explicit and implicit variables in information retrieval (IR) systems. Journal of the American Society for Information Science, 26(4).

Robertson, S. E. (1977). The probability ranking principle in IR. Journal of documentation, 33(4), 294.

Robertson, S. E., & Jones, K. S. (1976). Relevance weighting of search terms. Journal

of the American Society for Information Science, 27(3), 129.

Robertson, S. E., Rijsbergen, K. v., & Porter, M. F. (1980). Probabilistic models of indexing and searching. Proceedings of the 3rd annual ACM conference on Research and development in information retrieval.

Robertson, S. E., & Spärck Jones, K. (1976). Relevance weighting of search terms. Journal of the American Society for Information Science, 27(3), 129.

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M., & Gatford, M. (1995). Okapi at TREC-3. In Proceedings of the Third Text REtrieval Conference.

Rocchio, J. J. (1971). Relevance feedback in information retrieval. The SMART retrieval system: experiments in automatic document processing, 313-323.

Sakai, T. (2006). Evaluating evaluation metrics based on the bootstrap. Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval.

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

Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science, 41(4), 288.

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 620.

Sandhaus, E. (2008). The New York Times Annotated Corpus. Linguistic Data Consortium, Philadelphia.

Scott, J. (1988). Social network analysis. Sociology, 22(1), 109.

Shen, X., & Zhai, C. (2005). Active feedback in ad hoc information retrieval. Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval.

Silverstein, C., Henzinger, M., Marais, H., & Moricz, M. (1998). Analysis of a very large AltaVista query log. SRC Technical note, 14, 1998.

Silverstein, C., Marais, H., Henzinger, M., & Moricz, M. (1999). Analysis of a very large web search engine query log. ACM SIGIR Forum, 33(1), 6.

Singhal, A. (2001). Modern Information Retrieval: A Brief Overview. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering.

Sivic, J., Russell, B., Efros, A., Zisserman, A., & Freeman, W. (2005). Discovering object categories in image collections. International conference on computer vision.

Spärck Jones, K. (1971). Automatic Keyword Clawiscation for Information Retrieval. Proceedings of annual international ACM SIGIR conference on Research and development in information retrieval.

Sparck Jones, K. (1974). Automatic indexing. Journal of Documentation, 30(4), 393-432.

Steyvers, M., & Griffiths, T. (2007). Probabilistic topic models. Handbook of Latent Semantic Analysis, 424-440.

Tao, T., & Zhai, C. (2006). Regularized estimation of mixture models for robust pseudo-relevance feedback. Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval.

Tayebi, M., Hashemi, S., & Mohades, A. (2007). B2Rank: An Algorithm for Ranking Blogs Based on Behavioral Features. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence.

Teh, Y., Jordan, M., Beal, M., & Blei, D. (2006). Hierarchical dirichlet processes. Journal of the American Statistical Association, 101(476), 1566-1581.

Ter Hofstede, A., Proper, H., & van der Weide, T. (1996). Query formulation as an information retrieval problem. The Computer Journal, 39(4), 255-274.

Thomas, C. G., & Fischer, G. (1996). Using agents to improve the usability and usefulness of the World-Wide Web. Proceedings UM-96, Fifth International Conference on User Modeling.

Titov, I., & McDonald, R. (2008). Modeling online reviews with multi-grain topic models. Proceeding of the 17th international conference on World Wide Web.

Trotman, A. (2005). Learning to Rank. Information Retrieval, 8(3), 359.

Turtle, H., & Croft, W. (1991). Evaluation of an inference network-based retrieval

model. ACM Transactions on Information Systems (TOIS), 9(3), 187-222.

Vélez, B., Weiss, R., Sheldon, M. A., & Gifford, D. K. (1997). Fast and effective query. Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval.

Voorhees, E. (1985). The cluster hypothesis revisited. Proceedings of the 8th annual international ACM SIGIR conference on Research and development in information retrieval.

Voorhees, E. M., & Harman, D. (2000). Overview of the sixth text retrieval conference (TREC-6). Information Processing and Management, 36(1), 3-35.

Warin, M. (2004). Using WordNet and Semantic Similarity to Disambiguate an Ontology. Retrieved January, 25, 2008.

Wei, X., & Croft, W. B. (2006). LDA-based document models for ad-hoc retrieval. Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval.

Wen, J.-R., Nie, J.-Y., & Zhang, H.-J. (2001). Clustering user queries of a search engine. Proceedings of the 10th international World Wide Web Conference.

Wen, J., Nie, J., & Zhang, H. (2002). Query clustering using user logs. ACM Transactions on Information Systems, 20(1), 59-81.

White, R. W., Bailey, P., & Chen, L. (2009). Predicting User Interests from Contextual Information. Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval.

Witten, I. H., Moffat, A., & Bell, T. C. (1994). Managing gigabytes: compressing and indexing documents and images: Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Wong, S., & Yao, Y. (1989). A Probability Distribution Model for Information Retrieval. Information processing and management, 25(1), 39-53.

Xu, J., & Croft, W. (1996). Query expansion using local and global document analysis. Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval.

Xu, J., & Croft, W. B. (1999). Cluster-based language models for distributed retrieval.

Proceedings of the 22th annual international ACM SIGIR conference on Research and development in information retrieval.

Xu, J., & Croft, W. B. (2000). Improving the effectiveness of information retrieval with local context analysis. ACM Trans. Inf. Syst., 18(1), 79-112.

Yang, Y., Bansal, N., Dakka, W., Ipeirotis, P., Koudas, N., & Papadias, D. (2009). Query by document. Proceedings of the Second ACM International Conference on Web Search and Data Mining.

Yi, X., & Allan, J. (2009). A Comparative Study of Utilizing Topic Models for Information Retrieval. Proceedings of the 31th European Conference on Information Retrieval.

Zhai, C. (2002). Risk minimization and language modeling in text retrieval. Carnegie Mellon University.

Zhai, C. (2008). Statistical Language Models for Information Retrieval - A Critical Review.

Zhai, C., & Lafferty, J. (2001). Model-based feedback in the language modeling approach to information retrieval. Proceedings of the 10th International conference on Information and knowledge management

Zhai, C., & Lafferty, J. (2004). A Study of Smoothing Methods for Language Models Applied to Information Retrieval. ACM Transactions on Information Systems (TOIS), 22(2), 179-214.

Zhao, Q., Hoi, S. C. H., Liu, T.-Y., Bhowmick, S. S., Lyu, M. R., & Ma, W.-Y. (2006). Time-dependent semantic similarity measure of queries using historical click-through data. Proceedings of the 15th international conference on World Wide Web.

Zhou, D., Bian, J., Zheng, S., Zha, H., & Giles, C. (2008). Exploring social annotations for information retrieval. Proceeding of the 17th international conference on World Wide Web.

Zhou, Y. & Croft W. (2007) Query Performance Prediction in Web Search Environments. Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval.