Computational Community Interest for Ranking

Xiaozhong Liu
School of Information Studies
Syracuse University, Syracuse NY 13210
xliu12@syr.edu

Vadim von Brzeski
Yahoo! Inc,
Santa Clara, USA 95054
vadimv@yahooinc.com

ABSTRACT

Ranking documents with respect to users' information needs is a challenging task, due, in part, to the dynamic nature of users' interest with respect to a query, which can changes over time. In this paper, we propose an innovative method for characterizing the interests of a community of users at a specific point in time and for using this characterization to alter the ranking of documents retrieved for a query. By generating a community interest vector (CIV) for a given query, we measure the community interest by computing a score in a specific document or web page retrieved by the query. This score is based on a continuously updated set of recent (daily or past few hours) user-oriented text data. When applying our method in ranking Yahoo! Buzz results, the CIV score improves relevant results by 16% as determined by real-world user evaluation.

General Terms

Algorithms, Human Factors, Experimentation

Keywords

Information Retrieval, Ranking, Topic, User, Blog, Community Interest, LDA

1. INTRODUCTION

Ranking is a key step in Information Retrieval (IR) systems. All ranking algorithms work to find the most *important* documents and show them to users at the top of the search results.

Generally, existing ranking algorithms measure the "*importance*" of the document in the search results in several different ways, such like the distance between query and document in a high dimension vector space, probability of the document generating the query, social network popularity in the retrieved result and so on.

Two basic hypotheses are common in the existing algorithms: first, the query contains the key information for ranking, which provides hints used to discriminate the *important* results from others. Second, some unique features on the document or user side can help rank the results, such as citations, page links, or user behavior data.

There are also some limitations regarding these hypotheses. First, web queries tend to be short (Jansen, 1998; Silverstein, 1999) and the algorithms have relatively limited information on which to base their ranking. At the same time, some additional ranking information, such as links, citations or user behavior data can be biased. For instance, a blog posting getting a high number of citations or clicks may due to two different reasons: the content is interesting (it should get the high rank), or the blogger is popular in a certain community (the content may be pedestrian and does not deserve the high rank outside the local community). The content-free ranking algorithm will favor these postings no matter which scenario they belong to.

In this paper, we use "*community interest*" as an indicator to represent this *importance* score; namely, we compute a measure of the interest level in the global community in a specific retrieved document for a given query at a given time. Instead of employing a large number of users to make judgments as to what is interesting and what isn't, we use *user oriented text data* (such as daily or hourly blog postings or user selected news text) to represent users' interests, and the text is represented by different features. By using a popular topic-modeling algorithm (LDA), we discover *topics* of community interest in the user text data as probability distributions over the space of features. Each topic is then weighted by historical text data from the community. Finally we construct the CIV as a vector of weighted topics to represent the current interests of the community. For each document in the search results, we also infer a score (using the precomputed probability topic models) that is proportional to the level with which the community may be interested in this specific document given the query. This score, the CIV score, is then used for ranking the entire set of search results. In our algorithm, each user oriented text is viewed as an "agent" of user, and the real time topics of the text will be used to "vote" for the important documents over others.

Some terminologies mentioned in this paper list as following:

Term	*Definition*
User-oriented text	The text data generated by an end-user, such as blog postings, user selected news stories or news comments
Posting	A document generated by a user, an instance of user-oriented text
Protagonist	The main actor in the posting. A protagonist in this paper is not necessarily a person; instead, it is a user oriented entity from query-log
Topic	A (probability) distribution over a space of features
Community Interest	A list (vector) of threads of interests (topics) with respect to the target protagonist
Community Interest Vector	CIV is the weighted topics' score corresponding to the current (real-time) cognitive global community interest
CIV ranking	The ranking algorithm based on the real-time CIV inference score

Tab1. Terminology used in this paper

2. RELATED WORK

A number of techniques have been developed for ranking retrieved documents and web pages for a given query. The

classical method is content-based or query-dependent ranking, which is based on the similarity or probability of matching between query and target document. In web search, additional ranking information can be used, such as the hyperlinks between web pages, anchor text, user behavior data and the popularity of the page.

Content based ranking

Content based method rank the documents according to their relevance to a given search query. In vector-space content based ranking, the ranking score of a document with respect to a query is determined by its "distance" to the query vector (Kobayashi & Takeda, 2000), such like vector space model (Salton & Yang, 1973). In order to reduce the dimensionality of the vector and to represent "latent" word similarities, Latent Semantic Indexing, LSI, (Deerwester et al., 1999) is used to project the high dimensional word-document matrix into a lower one. Similarly, language models (Lafferty & Zhai, 2001) and probabilistic models (Joneset al., 1976) calculate the probability of a document generating the query and the probability of relevance based on the query and the document respectively. Related to our work, a topic based language model using LDA has been studied for ad-hoc information retrieval by (Azzopardi et al, 2004, Wei& Croft 2006). Recent studies combined existing ranking algorithms by machine learning to create new ranking functions trained by evaluation method, which is learning to rank (Trotman, 2005). In this approach, user interest and requirement are represented by query terms.

Linkage based ranking

In the WWW environment, the network structure of a hyperlinked network can be a rich source of information about the content of the pages, providing an effective means to understand it. The related ranking algorithms are like PageRank (Page et al., 1998) and HITS (Hypertext Induced Topic Selection) algorithm (Kleinberg, 1999), which are based on traditional citation analysis and social network analysis. Some more recent research combines these two approaches together and uses both content and hyperlinks to rank the search results. For instance (Haveliwala, 2003) worked with topic-sensitive PageRank, which created a list of PageRank vectors by using a set of representative topics in order to capture the context of each hyperlink. Similarly, a probabilistic model was used by (Richardson&Domingos, 2002) to generate PageRank score for each possible query term. In this approach, user interest is partially represented by network connected to the target document.

User behavior based ranking

User behavior data have been used and proved as an effective indicator for ranking. The relationship between implicit and explicit user data was studied by (Fox et al 2005), and two different Bayesian models were built to correlate different kinds of implicit measures and explicit relevance judgments for individual page visits and entire search session. 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 for ranking. Similarly, Agichtein et al (2006) incorporated noisy user behavior data into the search process, and the user data were used to train the ranking functions. In this approach, user interest is partially represented by statistical user behavior data.

In this paper, we are focusing on representing user interest from the topic distribution over user generated real-time text data, which is separated from the retrieved results. Instead of using statistics of user behavior data, we rank the documents in terms of the content of large amount of real time user generated text.

3. COMMUNITY INTEREST RANKING

In the Web 2.0 context, a user may generate different kinds of text data, such as blogs, selected news, or comments to express their opinion. We hypothesize that a large amount of user-oriented data can represent the overall opinion of the community. A simple example is the 2008 presidential election. As the following diagram shows, the number of blog postings about "Obama" and "McCain" changed over time (data from Yahoo! Buzz, http://buzz.yahoo.com, from 2008-10-11, before election, to 2008-11-10, after election).

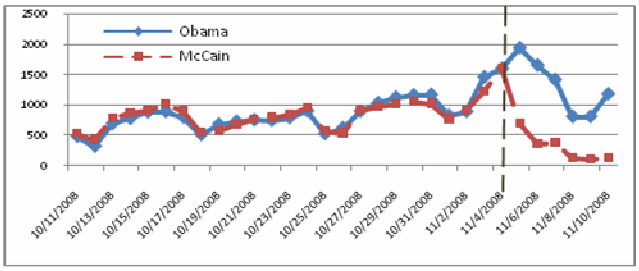

Fig 1. 2008 president candidates related blog postings

It is shown that before election day (2008-11-4), the numbers of postings about the two candidates are almost equal, but after the election, because of the results, the gap between the winner and the loser significant increased, representing the situation in the real world. Similar blog research about the 2004 election can be found in (Adamic & Glance, 2005).

In this paper, there are three central questions to answer:

1. *How can we accurately extract community interests via user oriented text data for a given "protagonist"?*
2. What are multiple interest threads of a protagonist, *and how can we weigh each interest (thread of discussion) to mirror the real world community's requirement?*
3. *How do we use this computational community interest to rank (or re-rank) the documents in the search result?*

The protagonist defined in our paper is the main actor (not necessarily a person) of a user generated posting. And the protagonist list is generated through query log. One protagonist can correspond to one query or multiple similar queries. Query similarity is well studied in different researches, such like (Wen et al, 2002, Baeza-Yates et al., 2004).

In this section, we will describe our method and try to answer the aforementioned questions.

3.1 Community Interest Extraction

If we index user oriented text by protagonist, and each protagonist represents one or several similar hot (i.e. high frequency) queries identified in query logs, then for each protagonist, we can collect a number of user oriented postings for a period of time (e.g. today, or past few hours). We call this collection of postings the "Current Protagonist Collection" (CPC). When the number of postings increases, the representability of this collection (and of community) also increases.

We define the community's interest toward each protagonist as a vector – the *Community Interest Vector* (CIV), and each component of the vector represents a (normalized) "*topic*" related to the target protagonist. This interest vector may change in two different ways over time:

1. *Vector space change – Since each dimension in the vector represents a topic of interest about the target protagonist, 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 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 protagonist "Obama" as an example, when n = 3, the CIV of "Obama" for the 1st of Aug, Oct, and Dec of 2008 may look like following:

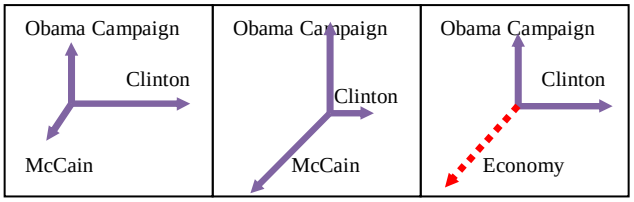

Fig 2. Three dimensional topic space change

In August, the community was interested in three different topics about "Obama": 1. Obama's campaign 2.the relationship between Obama and Clinton 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 at that time. In October, these topics may still exist, but the weights of first and third topic have increased, while the weight of the second has decreased. In December, after the election, the third topic is replaced by the "economy", and the weight of each topic also changes.

Our algorithm uses real-time user oriented text data as the corpus to extract and weight CIV, and each item in CIV represents a current topic, which is a probability distribution over features.

Fig 3 shows how CIV algorithm ranks the documents generally. In the simplest way, each query corresponds to one protagonist. The query is sent to both indexed documents and user oriented text collections. The algorithm will extract the CIV from current protagonist collection based on topic extraction and topic weighting modules, and finally the CIV will rank the candidate retrieved results against current topic distribution.

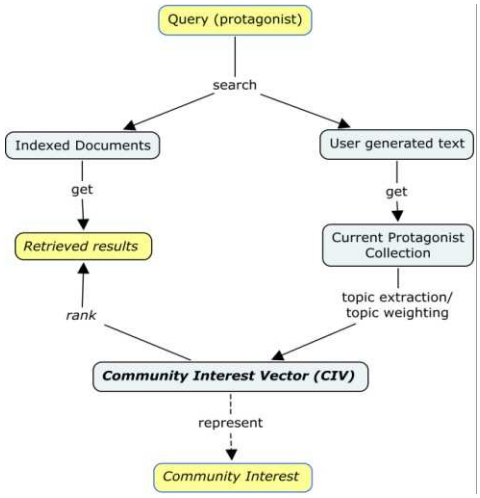

Fig3. CIV algorithm

3.2 Topic Extraction with LDA

We hypothesize that the postings in the CPC incorporate a fixed number of *latent topics* and we proceed to extract these topics. A topic in our model is a probability distribution over features. There are various techniques to perform this topic modeling step, and we chose an off-the-shelf public domain algorithm called Latent Dirichlet Allocation (LDA), (Blei, Ng, & Jordan, 2003). In a nutshell, LDA is similar to *probabilistic Latent Semantic Analysis*(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.

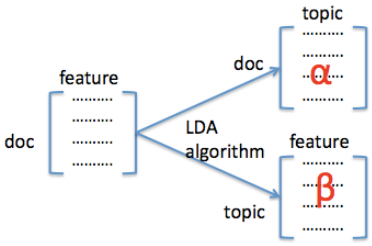

Fig 4. LDA topic extraction

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. By using LDA, the document-feature matrix can produce two different matrices: α contains the document (posting) – topic probability distributions, i.e. each row represents the probability of topic given the posting *P (topic | posting)*. 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 ϕ_z for each topic z from a Dirichlet distribution with hyperparameters β. ϕ_z models the relative frequencies of features in topic z.
2) For each document d, draw parameters for a multinomial distribution θ_d from a Dirichlet distribution with hyperparameter α. θ_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 θ_d.
b) Given z_n, draw a feature (word) w from the multinomial distribution ϕ_{zn}.

An example of an LDA result from a user oriented text collection is shown as following:

Swimming Topic		Russia War Topic		Gymnast Topic	
Wiki:Michael_Phelps	0.024279	georgia	0.008965	Liukin	0.011639
Wiki:Phelps	0.017913	Wiki:Russia	0.007939	Wiki:Nastia_Liukin	0.011465
Swimming	0.011785	spanish	0.00598	Wiki:Gymnast	0.010945
Wiki:Gold_medal	0.011387	russia	0.005513	Wiki:Shawn_Johnson	0.009469
Wiki:Swimming	0.010114	war	0.005047	Gymnastics	0.007647
200m	0.009398	Wiki: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	Wiki:George_W._Bush	0.002062	Young	0.003567
Wiki: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	TVS:nastia	0.003307
rebecca_soni	0.003668	did	0.001782	Old	0.00322
100m	0.003509	Wiki:Spain	0.001782	Father	0.00322
Swimmer	0.003509	iraq	0.001688	chinese_gymnasts	0.00322
Wiki:Medley_swim	0.003509	oil	0.001595	Years	0.003133
Wiki:Ryan_Lochte	0.003509	russias	0.001595	Wiki:Uneven_bars	0.00295
Jones	0.003271	south	0.001502	Wiki:Gymnastics	0.002786

Tab2. Three topic-feature distribution example

The above table shows three sample topics extracted from the 2008-08-11 blog posting collection (1086 postings, number of topic = 30, protagonist = "Olympic"). Each topic is represented by features (bag-of-words + entities + Wikipedia ID – see a detailed description in the next section), and the probability of the feature given topic *P(feature | topic)*. We printed the top 20 features of each sample topic.

Based on the topic-feature probability distribution, we can use the learned LDA model to infer the topic distribution in a new document. Given a new unseen document, by inverting the LDA generative process, we can obtain the topic probability distribution in the new document. Each dimension represents the relative frequency (probability) of each topic in the document belongs:

$$TV(doc_x) = \{P(topic_1 \mid doc_x), P(topic_2 \mid doc_x) ..... P(topic_n \mid doc_x)\}$$

$TV(doc_x)$ 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 Feature Space

In any information retrieval and text mining system, features are important as the units, which represent the indexed document and corpus. However, compared with traditional retrieval systems and web search engines, the quality of user-generated text (such as blog postings) is low, due to spelling mistakes, grammatical mistakes, and spoken language expressions. These issues pose a challenge to the performance of existing NLP, IR, and mining algorithms. Furthermore, users tend to use different terms and phrases to express the same thing, which not only increases the dimensions of the feature space, but also misleadingly divides the same feature into different ones. Last but not least, we find users sometimes write people's name or locations without capitalization, and this behavior removes one of the most important features for entity recognition algorithms.

In order to solve the aforementioned problems, we need to design an algorithm to:

1. *Recognize all the possible entities from user oriented text data, even in the presence of grammatical and spelling mistakes.*
2. *Project the low quality entities into a "clean" Wikipedia ID, which has the closest semantic distance with the entity.*

In our experiment, we find using the correct type of features to represent user generated text can improve the system performance. However, this section is somehow independent from our algorithm. So if you are not interest in the feature representation, please jump to section 3.4.

Entity recognition

We employed Contextual Shortcuts Platform (von Brzeski et al., 2007) to extract entities from text. Contextual Shortcuts uses a combination of dictionary and machine learning approaches to determine the set of most relevant named entities and keywords (concepts) in a piece of text. Dictionaries themselves consist of editorially constructed lists of named entities (e.g. persons, places, organizations, etc., organized in a shallow taxonomy) and an automatically generated list of concepts derived from query terms found in web search logs. In fact, the automatically generated list is much larger than the set of editorially derived terms. Because some existing popular terms in query logs may also contain the same errors as user oriented text data, such as "*barrack obama*" (spelling mistake), this entity extraction algorithm can extract some low quality entities. We can project such entities to high quality semantic features in next step. Furthermore, the Contextual Shortcuts Platform performs entity disambiguation and ranks the entities discovered in a piece of text according to their relevance to the main topic of the text, as well as their interestingness to the broad user community (Irmaket al., 2009).

Finding the candidate Wikipedia IDs

Bloggers tend to use abbreviations and ambiguous entities; readers can understand the meaning by the context. An example could be the following sentences containing the entity "Detroit":

1. *Detroit won the game last night. (protagonist = "NBA")*
2. *Detroit will be better on the ice this season. (protagonist = "NHL")*
3. *Obama will win Detroit. (protagonist = "Presidential Election")*

When considering the context (such as the query context or blog context) of the ambiguous entity, we can figure out the real concept that the entity corresponds to (1. "Detroit Pistons"; 2. "Detroit Red Wings"; 3. "Detroit, Michigan"). We designed the semantic similarity algorithm based on the work of (Cilibrasi &

Vitanyi, 2007) to automatically identify the closest concept from the Wikipedia database.

Cilibrasi & Vitanyi (2007) computed the normalized semantic relatedness between two entities using the Google distance. In our experiment, we first search for the extracted (ambiguous) entity in a Wikipedia resolver component, which returns a list of Wikipedia IDs given a named entity or concept (e.g. Detroit_Pistons or Detroit_Red_Wings from "Detroit"). The Wiki resolver was built by analyzing the link structure of Wikipedia in order to associate anchor text with Wikipedia page names. It uses query terms in web search logs in order to associate queries (e.g. named entities) with Wikipedia page names, and it also uses Wikipedia's editorially created redirect pages to associate those page names with Wikipedia pages. Finally, the above associations are merged into one final score mapping a query or named entity into one or more Wikipedia IDs. We then compute the Google distance between the entity and each Wikipedia concept (ID) in the context of protagonist (because we indexed the blog by protagonist) by the following formula.

$$G\_dis(entity, wiki \mid P) = \frac{Max\{\log C(entity, P), \log C(wiki, P)\} - \log C(entity, wiki, P)}{\log M - Min\{\log C(entity_1, P), \log C(entity_2, P)\}} \quad (1)$$

P is the target protagonist; C is the count of results returned from Google general web search; M is the total number of web pages indexed by Google; *G_dist* is the normalized Google distance between the Wikipedia ID and entity, ranging from 0 to 1 (0 means semantic identity, 1 means no semantic relatedness). In this way, we can find the closest semantic concept feature to replace the entity feature in the protagonist context (for instance, when protagonist is "NBA", the closest concept of "Detroit" is "Detroit Pistons"). Since we built the protagonist collection directly from query logs and used it later for ranking, the concept feature will logically help us improve the topic model learning as well as the ranking performance.

The final feature space is this combination of terms, extracted entities, and Wikipedia IDs.

3.4 Building the Community Interest Vector

After we generated the topic model from the CPC, it is very important to weight each topic. 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 we found through our 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) could be judged as a protagonist specific stopword list.*
2. *Hot topic: there are two types of hot topics for the community; first, a topic in which the community is continuously and increasingly focused, and second, a topic related to breaking news surrounding the protagonist, which is of great interest in the community.*
3. *Diminishing topic: the topic is no longer popular for community; the community's interest is shifting to other topic(s).*
4. *Regular topic: we cannot tell the popularity of the topic from historical data.*

We used historical data (past few hours or days) to classify topic type and compute the weight of each topic for the target protagonist. The most straightforward method is to compute a list of topic models for each corpus 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, we used the learned CPC topic model to infer the topics in the historical protagonist corpuses. The algorithm is as follows in Fig5.

As mentioned above, from the LDA model, we obtained two probability distributions: α - the probability of topic given the posting *P (topic | posting);* and β - the probability of each feature given the topic *P (feature | topic).* Based on these distributions, we compute the "*Current_topic_score[k]*" by summing the posting vectors from α. We also run the model against historical data (past n days or hours, n corpuses, worth of user oriented documents for the protagonist) and infer the topic distributions α in the historical data. Because the LDA model was build using the CPC, the historical postings can be viewed as unseen data. For each document, the inference result is:

$$TV(doc_x) = \{P(topic_1 \mid doc_x), P(topic_2 \mid doc_x) ..... P(topic_n \mid doc_x)\} \quad (2)$$

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 on the past i[th]

For each protagonist

```
Training_CPC_topic_model;          //k topics
Current_topic_score[k] = Σdocument_topic_dist[k];
//compute the CPC topic score by summing each doc-topic distribution in CPC
Compute History_topic_score[n][k]
//Inferring past n days (or hours) topic distribution based on CPC 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, p is the "bonus" parameter
    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
   Else
CIV[j] = Current_topic_score[j];
        //Regular topic
```

Fig 5. CIV building algorithm

day (or hour), the community (represented by the user oriented corpus) is interested in topic *k*. By comparing the mean and the standard deviation of specific topics' scores for a window of the past n days (or hours), we can decide if the topic is a "hot topic", "diminishing topic", or "regular topic" as shown in the algorithm.

$$CIV[j] = \begin{cases} b \cdot Current\_topic\_score[j] \cdot \dfrac{Current\_topic\_score[j]}{history\_topic\_score[i][j]} & hot-topic \\ p \cdot Current\_topic\_score[j] \cdot \dfrac{Current\_topic\_score[j]}{history\_topic\_score[i][j]} & diminishing-topic \\ Current\_topic\_score[j] & regular-topic \\ p' \cdot Current\_topic\_score[j] & background-topic \end{cases} \quad (3)$$

The hot-topic and diminishing-topic CIV scores were adjusted by the change rate of current topic score and the mean of the historical topic scores; a bonus parameter (*b, b>1*) and penalize parameter (*p, p<1*) were used in our algorithm to update the topic weight. In our experiments, *b = 1.2*, while *p = 0.8*. Because we identify the topic category by mean and standard deviation, the change rate of a hot-topic is always > 1 and the change rate of a diminishing-topic is always < 1. In our experiments, we also found that some topic's mean probability score is significant larger than all other topics' scores (at least 5 times larger) – we define these topics as "background topics", and penalize these topics' weights by p' = 0.2 (penalize parameter of background-topic). The background topic is mainly composed by a list of general and domain specific stopwords (for instance protagonist = "Obama", the stopwords can be "Obama" and "president"). Even though the background topics' weights are large in all the corpuses, these topics are harmful for community interest based ranking. (As shown in formula (3))

The following diagrams (Fig 6 & 7) are examples of CIV topic weighting. The protagonist is "Obama", and experiment time is Nov 5th 2008, one day after the 44th president election, the training corpus is Yahoo! Buzz postings (we will discuss the data in next section) and corpus size is 1,491 user generated postings. We show the highest weighted "Hot topic", which can be summarized as "Obama wins the election", and whose top features are "Barack_Obama", "Election", "African_American", "victory", "Victory_Records" and "first_black_president". We also show the lowest weighted "Diminishing topic", which can be characterized as "Sarah Palin and Hillary Clinton", with top features like "Sarah Palin", "sarah", "palin" "Hillary clinton", "newsweek", "club", "cloth".

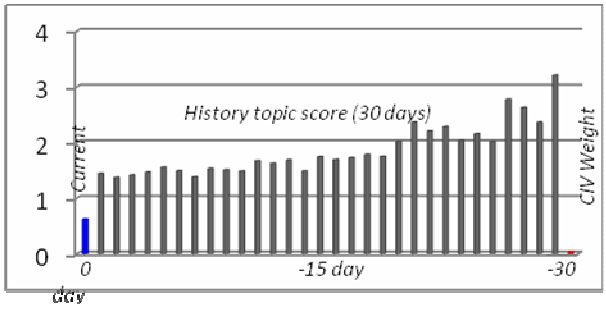

Fig 6. Nov 5th, Diminishing topic: "Sarah Palin & Hillary Clinton"

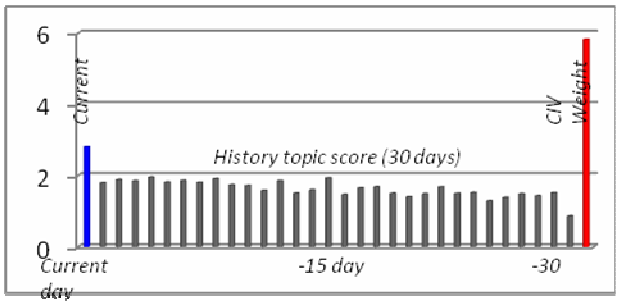

Fig 7. Nov 5th, Hot topic: "Obama win president election"

In the example(Fig 6 and 7), we compute the topic-feature distribution using the current "Nov 5th" corpus about "Obama", and then use it to infer topic distributions in the past 30 days (from Oct 5th to Nov 4th). By computing the mean and the standard deviation of the topic probability scores, we can identify hot and diminishing topics by their final weights in the CIV. In the diagram, the first bar on the left is the initial (current) topic weight, and the last bar on the right is the final adjusted weight of the topic in the CIV.

3.5 Ranking

When a query is equal to or is similar to the protagonist, we can bias the ranking result by using the current Community Interest Vector. For any given retrieved document collection *R (doc1, doc2… docK)*, based on our topic model (α, *topic-feature distribution*), we can infer the topic distribution of each document in the search results as mentioned earlier. Because the topic vector of each document in the search results is in the same vector space as CIV, we can compute the final document interest score by cosine vector similarity:

$$ranking\_score(doc_x) = Sim(CIV, TV(doc_x)) = \frac{\sum_{i=1}^{n} CIV(topic_i) \cdot TV(doc_x, topic_i)}{\sqrt{\sum_{i=1}^{n} CIV(topic_i)^2} \cdot \sqrt{\sum_{i=1}^{n} TV(doc_x, topic_i)^2}} \quad (4)$$

Since the CIV represents the community's interest with respect to each protagonist, the final ranking score can be viewed as a *pseudo-voting* based ranking, where the user oriented text data serves as a proxy for the votes. Thus, the ranking score can represent *P(interest | doc)*, the probability that a community is interested in a given document.

4. EXPERIMENTS

4.1 Data

In this paper, we focus on computational community interest, and we need to use real-time user oriented text data to represent the community's interests. We chose Yahoo! Buzz data (http://buzz.yahoo.com) for the following reasons. First, this is a user oriented text dataset (mainly in the news domain), which primarily includes two parts: user selected news stories and user oriented news comments. Second, a user may copy and paste from other news services (like CNN.com), but they tend to select only specific parts of the news instead of the whole story. The selective

sentences or passages have higher probability of being interesting to the global community, since background context information is filtered out. This is also beneficial for our interest extraction algorithm and ranking. Third, compared with blog data, Yahoo! Buzz focuses more on news instead of social network communications, which facilitates news based ranking and user evaluation. Finally, each Buzz posting contains a time stamp that can be used for corpus selection.

We selected 129 hot queries from news search engine query logs, and used those query terms and entities directly as the protagonist to search and index Buzz postings. Here, we do not use a protagonist detection algorithm (an algorithm which attempts to confirm that a given posting is actually *about* the target protagonist) for two reasons. First, we want to get enough text data for building the community interest topic model (corpus size is important in obtaining a useful model); protagonist detection algorithms may filter a large percentage of postings. Second, some existing protagonist algorithms are very time consuming, and we want the ranking algorithm to be used in an online information retrieval system. However, using a query directly as a protagonist will bring in some noisy data (see below).

From Oct to Dec 2008, we indexed 274,400 postings with an average length of 769 characters. The postings were indexed by protagonist (the entity from query log), stemmed words, entities and Wikipedia concepts as well as the published time stamp. As mentioned earlier, we used Contextual Shortcuts as the named entity recognition algorithm to find all the entities within the postings, and then, for each entity, we find the candidate Wikipedia concepts for each ambiguous entity by using the Wikipedia resolver component mentioned earlier. Finally, we computed the Normalized Google Distance (NGD) in the context of the protagonist (equation 4), and replace the entity feature with the Wikipedia concept feature if possible.

One posting may correspond to several different protagonists. The index module was checking for new postings from the user community about these 129 protagonists in real-time during the experiment, and the attached publish time stamp was used to filter the training ("current") and historical corpuses for topic extraction and topic weighting.

4.2 Topic Modeling (Training)

We select the training corpus as follows:

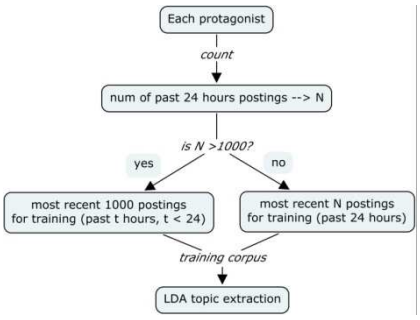

Fig 8. Workflow to identify training corpus

For each protagonist, if there were more than 1000 postings published in the past 24 hours, we capped the training corpus (CPC) size at 1000, which represents community's interest toward the protagonist for the past *t* hours ($t < 24$). Otherwise, we will use the past 24 hours worth of postings for LDA model construction.

After comparing several different values for the "number of topics" parameter K in LDA, we fixed K at 30, as the extracted topics should be neither too general nor too specific. We set the LDA parameters setting as $\alpha = 50/K$ and $\beta = 0.1$. In the experiment, we find the topic number and β value significantly influenced the validity of CIV and ranking performance. Generally the lower β, the sparser the topics will be, which means the model prefers to assign few terms to each topic (Heinrich, 2005). Meanwhile, the number of topic K defines how many cognitive dimensions we need to define for each CIV. Unfortunately, we cannot guarantee that the experimental parameter value is the optimized as we didn't have enough evaluation resources (we will mention that in next section) to compare the performance across different parameter settings. Instead, in this paper, we used intuitive best values for this experiment by comparing different parameter setting by ourselves.

4.3 CIV Weighting

To build the final Community Interest Vector (CIV), we need to infer the corpus-topic distributions for the past *n* days (or hours) and use trend analysis to weight each topic.

For each protagonist, we used a corpus of size *(m)*, as in the previous LDA training step. In the historical posting collection about a target protagonist, we composed *h* corpuses (in experiment $h = 30$) ordered by publish time, each corpus containing *m* postings the same as the training corpus. Because daily or hourly variation of corpus size may be large and we don't want the inference performance dropped significantly due to the corpuses' unbalance. By analyzing the inferred topic probability scores (component by component) across different days' corpuses, we computed the final topic weight as the CIV:

$$CIV(protagonist) = [W\text{-}topic_1, W\text{-}topic_2 \ldots W\text{-}topic_t] \quad (5)$$

The CIV reflects the current community interest toward the protagonist. We will use this vector directly for ranking. During the experimental period, we computed a CIV for each protagonist 4 times a day.

4.4 Ranking and Duplicate Detection

We use the same Yahoo! Buzz source for the ranking test. We sent each protagonist as query to Buzz search, which returned the ranked Buzz postings for the recent three days. Each retrieved document was treated as an unseen document and we inferred the document-topic distribution based on the existing LDA topic model for the target protagonist. The rank score for a document was calculated using equation4.

However, in the ranked result, we find some duplicate results for two reasons:

Content duplication: the content is virtually identical, same verbiage, but maybe in different word order; talks about the same event.

Topic duplication: the words may different, but the topic distribution is similar and it talks about the same event

We attempt to detect and remove duplicate news stories from the result. For content duplication, bi-gram fuzzy edit distance was used to identify duplicate documents. If the fuzzy similarity of the

title and the first paragraph was larger than a threshold (in the experiment it was 0.8), the duplicate document will be removed from the result. For topic duplication, the inferred document-topic vector cosine similarity was computed between documents; if the topic similarity is larger than a threshold (in the experiment it was0.8), the duplicate document will be removed.

5. EVALUATION

The evaluation of a ranking algorithm is difficult, especially for our real-time ranking task, which cannot employ existing test collections such like 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. And *Normalized Discount Cumulative Gain (NDCG)* (Järvelin & Kekäläinen, 2002) works when user graded relevance data is available.

For this paper, the most important contribution is to capture the dynamic community interest since community interest may change from time to time. As a result, we have to conduct a real-world evaluation based on selected protagonists over a period of time. We focus on the "Interesting & hot rate"-at-document-n as well as the "Not interesting or not relevant rate"-at-document-n.

We set up a preliminary evaluation with five real readers for a period of 5 days (Nov 11, 12, 13, 14, 17 2008) intended to test the concept and may serve for future large scale evaluations. Nine queries were randomly chosen in the evaluation (from top frequent query log of the first two weeks of November 2008) as following table shows:

Query:	Average Training Size	Average CIV covered time Interval
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

Tab 3. Nine queries for evaluation.

On the five evaluation days, we constructed a topic model every day at 14:00PM and users accessed our evaluation system on 15:00PM to make their judgments. The above table shows the average number of training documents for LDA topic extraction for each query (protagonist). "Obama" and "Economy" are the popular protagonists during that time, which exceeded the threshold, and we used only 1000 most recent Buzz postings for training (the 1000 postings covered 18.3 and 16.1 hours community interests respectively).

In the evaluation system, after logging in, the judges were required to click nine queries one by one and read the top 5 ranked documents in two collections each: *Yahoo! Buzz ranked results* and *CIV algorithm ranked* results. Two different ranked set were randomly presented to user. After reading each ranked search result, users were asked to grade one of the following options about this document:

- *"Interesting and Hot" = This document is directly relevant to the given query and it is about something currently interesting and hot in the news.*
- *"Mildly interesting" = This document is relevant to the given query, BUT it is about something no longer in the news.*
- *"Not interesting or Not relevant" = This document is relevant to the query, but is completely not interesting or new, or it is not directly related to the topic.*
- *"Duplicate" = This document talks about the same event as another document in the same subset.*

For 5 five days for 5 users and 9 queries corresponding to top 5 documents for each algorithm, there were a total of 2,250 valid evaluation results collected. We first employed the idea of *Precision-at-document-n*, and averaged rates of above four categories for each ranking method. The results are shown in the following table:

		Interesting and Hot	Mildly Interesting	Not interesting or Not relevant	Duplicate	Interest increase	Not Interest increase
11/17/2008 Monday	CIV	46.72%	20.44%	22.63%	8.76%	6.57%	-16.79%
	BUZZ	40.15%	18.25%	39.42%	0.73%		
11/14/2008 Friday	CIV	60.74%	20.74%	10.37%	8.15%	15.56%	-25.93%
	BUZZ	45.19%	17.78%	36.30%	0.74%		
11/13/2008 Thursday	CIV	67.54%	18.86%	7.46%	6.14%	25.88%	-26.75%
	BUZZ	41.67%	23.25%	34.21%	0.44%		
11/12/2008 Wednesday	CIV	60.22%	20.44%	12.71%	6.63%	15.47%	-18.78%
	BUZZ	44.75%	23.20%	31.49%	0.55%		
11/11/2008 Tuesday	CIV	53.74%	25.11%	16.30%	3.96%	15.42%	-14.98%
	BUZZ	38.33%	28.19%	31.28%	2.20%		
ALL	CIV	58.48%	21.26%	13.44%	6.39%	16.74%	-20.59%
	BUZZ	41.74%	22.91%	34.03%	0.99%		

Table 4. Precision-at-document-n Evaluation results.

In the evaluation results, we focus on two questions: whether the CIV ranking can improve "Interesting & Hot" rate; and whether the CIV ranking can decrease "Not interesting or Not relevant" rate. In the following diagram, we present these answers in a clearer way:

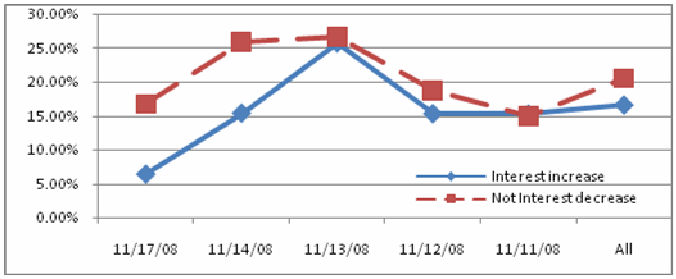

Fig 8. Comparing CIV ranking with existing ranking method

In fig 8, the lower line shows that for five days, the "Interesting & Hot" rate increased 16.74% on average as compared to the existing Yahoo! Buzz ranking algorithm. The upper line shows that the "Not interesting or Not relevant" rate decreased 20.59% on average. On Monday, 11/11/2008, the "Interesting & Hot" rate increased by only 6.57%, on all the other days it increased by more than 15%.

Secondly, NDCG evaluation was used to process the graded relevance judgments. We set the "Interesting & Hot" relevance rate = 3, as user values these results significantly better than other results. Meanwhile, "Mildly interesting" rate = 1, "Not interesting or Not relevant" rate = 0 and "Duplicate" rate = 0.

Based on these definitions, we compute the average NDCG@3 and NDCG@5 across test queries based on (Järvelin & Kekäläinen, 2002). The results are shown in the following table with significant test:

	NDCG@3	NDCG@5	Significant test
BUZZ	0.5740	0.7597	p<0.05 significant
CIV	0.8619	0.8874	p<0.1 significant

Table 5. NDCG Evaluation results.

From the evaluation results, we find that the CIV ranking algorithm significantly increased the ranking performance (for both Precision-at-document-n and NDCG) compared with an existing popular search engine ranking algorithm.

6. DISCUSSION

The preliminary evaluation shows that the global community interest is a good indicator for IR ranking. However, in our experiments, we still found some limitations in this algorithm.

First, some queries (or protagonists) are ambiguous, and LDA cannot directly help us separate the topics semantically for ranking. This can be a problem of the polysemy effect (Sparck Jones, 1972). For instance "Georgia" is an ambiguous query, which can represent "a state in the United States" or "a country". Building a CIV for "Georgia" (in Oct 2008) is risky, as it will mix the "US presidential election" and the "Georgia war" topics into the same vector space, and we may need a word sense disambiguation algorithm to solve his problem.

Second, we did not use a protagonist detection/verification algorithm to better clean the training corpus in the experiment, resulting in some noisy data leaking into the training corpus. For instance, the word "Obama" shown in one posting does not necessarily mean that "Obama" is the protagonist of the posting. In our experiment, we did not implement the protagonist detection/verification because of the data (training) size problem. In future research, we need to collect more (user-oriented) text data and filter a higher quality training corpus for topic extraction.

Last, we find CIV ranking algorithm generates more topic duplicate results (shows in Tab 4), even though we used duplicate detection. A possible reason is the topic distributions among the top ranked results are similar in our algorithm as they are all extracted from the same corpus. Better duplicate detection algorithm should be used in the future work to reduce the duplicate rate.

Another finding was that the training and historical corpus size is important for the CIV ranking algorithm. An example of this is in the ranking performance on Monday 11/11/2008; it is lower than the other weekdays because users generated less text data (fewer postings) over the prior weekend, and we thus obtained fewer postings for training for community interest extraction.

In future evaluations, we will be targeting a larger scale user study with the goals of deciding an ideal training corpus, best feature types, and the optimized training and weighting parameter settings. Meanwhile, we hope to compare the CIV algorithm with the existing specific ranking algorithm individually. We will also develop the ranking algorithm with more sophisticated techniques.

We are currently working on Community Interest Language Model (CILM).

In summary, community interest modeling with the real world user oriented text data is an effective method for mirroring real world community from a cognitive perspective. By weighting each topic extracted from query driven time sensitive corpus, we can measure the degree of interest, namely, interest based ranking, which is differ from relevance ranking. The community interest vector in our experiments demonstrates as an effective way to rank retrieved results.

7. REFERENCES

[1] 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.

[2] 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

[3] Anh, V. N., &Moffat, A. (2002). Improved retrieval effectiveness through impact transformation. Paper presented at the ACM International Conference Proceeding Series.

[4] Azzopardi, L., Girolami, M and van Rijsbergen, C.J. (2004).Topic Based Language Models for ad hoc Information Retrieval. In proceedings of the International Joint Conference on Neural Networks, Budapest, Hungary

[5] Bernard J. Jansen, A. S., Judy Bateman, Tefko Saracevic. (1998). Real life information retrieval: a study of user queries on the Web. Paper presented at the ACM SIGIR.

[6] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. The Journal of Machine Learning Research, 3.

[7] von Brzeski, V., Irmak, U., &Kraft, R. (2007). Leveraging context in user-centric entity detection systems.Conference on Information and Knowledge Management. ACM. pp 691-700

[8] Cilibrasi, R. L., & Vitanyi, P. M. B. (2007). The Google Similarity Distance. IEEE Transactions on Knowledge and Data Engineering, 19(3).

[9] Craig Silverstein, H. M., Monika Henzinger, Michael Moricz. (1999). Analysis of a very large web search engine query log. Paper presented at the ACM SIGIR.

[10] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1999). Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41(6).

[11] Fox S., KarnawatK., MydlandM., DumaisS. & White T., (2005). Evaluating implicit measures to improve web search, ACM Transactions on Information Systems (TOIS), v.23 n.2, pp147-168

[12] Haveliwala, T. H. (2003). Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. IEEE Transactions on Knowledge and Data Engineering, 15(4).

[13] Heinrich, G. (2005). Parameter estimation for text analysis o. Document Number)

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

[15] Irmak, U., von Brzeski, V, & Kraft, R. (2009). Contextual Ranking of Keywords Using Click Data. Paper presented at the IEEE International Conference on Data Engineering.

[16] Järvelin K. & Kekäläinen J., Cumulated gain-based evaluation of IR techniques, ACM Transactions on Information Systems (TOIS), v.20 n.4, p.422-446, October 2002

[17] Joachims, T. (2002). Optimizing search engines using clickthrough data, Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, Edmonton, Alberta, Canada

[18] Jones, K. S., Walker, S., & Robertson, S. E. (1976). A probabilistic model of information retrieval: development and status.

[19] Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. Journal of the ACM, 46(5).

[20] Kobayashi, M., & Takeda, K. (2000). Information retrieval on the web. ACM Computing Surveys (CSUR), 32(2).

[21] 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.

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

[23] Richardson, M., & Domingos, P. (2002). The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank.

[24] Salton, G., & Yang, C. S. (1973). On the Specification of Term Values in Automatic Indexing. Journal of Documentation, 29(4).

[25] Sparck Jones, K. (1972). A statistical interpretation of term specificity and its applications in retrieval. Journal of Documentation, 28.

[26] 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, Seattle, Washington, USA

[27] Wen J., Nie J. & Zhang H., (2002). Query Clustering Using User Logs, ACM Transactions on Information Systems, 20(1), pp 59-81