UIC at TREC 2006 Blog Track

Wei Zhang
Department of Computer Science
University of Illinois at Chicago
wzhang@cs.uic.edu

Clement Yu
Department of Computer Science
University of Illinois at Chicago
yu@cs.uic.edu

ABSTRACT
We developed a two-step approach that finds relevant blog documents containing opinioned content for a given query topic. The first step, retrieval step, is to find documents relevant to the query. The second step, opinion identification step, is to find the documents containing opinions within the scope of the document set from the retrieval step. In the retrieval step, we try to improve the retrieval effectiveness by retrieving based on concepts, and doing query expansion using pseudo feedback, Wikipedia feedback and web feedback. In the opinion identification step, we train a sentence classifier using subjective sentences (opinioned) and objective sentences (non-opinioned), which are relevant to a query topic. This classifier labels each sentence in a given document as either subjective or objective. A document containing subjective sentences relating to the query is finally labeled as an opinioned relevant document (ORD). We tried two strategies to rank the ORDs that became two submitted runs.

1. INTRODUCTION
A blog document is said to be an opinioned document regarding a query topic, if this document is related to the query, and it contains user opinions about the query, no matter the opinions are positive or negative. It also does not matter if the opinions are from the blog article or from other people's comments. We consider this opinioned document retrieval as a two-step procedure. The first step is to retrieve the documents relevant to the query. This is an Information Retrieval (IR) process. We applied concept identification, query expansion techniques in this step to improve the effectiveness. The second step is to find the opinions relevant to the query in the retrieved documents, so that the document can be claimed as an opinioned document for the query. We look for the subjective sentences in a retrieved document. These sentences express opinions. Then the subjective sentences that are relevant to the query are extracted from the subjective sentence set. We call such sentences the *opinioned relevant sentences*. Given a query, we train a sentence classifier with a subjective sentence training set and an objective sentence training set, both of which are related to the query. This classifier is used to classify every sentence in a document as either subjective or objective. When a subjective sentence, *s*, is found, a group of (2n+1) sentences is generated by including *s*, n sentences prior to it, and n sentences after it. *S* is considered as a opinioned relevant sentence if a certain number of query terms are found in this sentence group.

2. QUERY PROCESSING
The “title only” run only utilizes a few terms in the query titles. These terms are not enough to achieve high-quality retrieval results. In order to improve the retrieval effectiveness, two query pre-processing methods are introduced. The first one is concept identification and concept based retrieval accordingly, which finds concepts from the original query and then retrieve documents based on the identified concepts. The other one is the query expansion. More relevant terms are collected by using methods, such as local pseudo feedback, Wikipedia and web-based feedback, to expand the original query.

2.1 CONCEPT IDENTIFICATION
We define a concept in a query as either a multi-word phrase consisting of adjacent query words, or a single query word that can not form a phrase with its adjacent words. Four types of concepts are defined with different significance power. They are proper nouns, dictionary phrases, simple phrases and complex phrases. Proper nouns are the noun phrases referring to people, place, event, organization, position, or other particular things. A dictionary phrase is a phrase that has an entry in a dictionary. Proper noun can be considered as a special type of dictionary phrase. A simple phrase is a 2-word noun phrase, which is grammatically valid but not having a dictionary entry, e.g. “small car”. A complex phrase is a grammatically valid noun phrase having 3 or more words but not having a dictionary entry. We developed an algorithm that combines several

tools to identify concepts in a query. We use Minipar, WordNet, and Wikipedia [3] for proper noun and dictionary phrase identification. Collins Parser is used to find the simple phrase and complex phrase. Web search engine (Google) is used to provide statistical information for phrase verification and selection purpose. The algorithm detail can be found in [11]

2.2 QUERY EXPANSION USING LOCAL PSEUDO FEEDBACK

In local pseudo feedback method, the original query is used to retrieve a ranked document list from a document collection. The terms that are highly correlated to the query terms in the top n documents of the list are returned as the expanded words. The assumption is that the top ranked documents in the list should be relevant to the query, and the terms co-occurred with the query terms should be related to the query terms. In our implementation, the term correlation calculation is similar to the one described in [1]. An extracted term is assigned a weight based on its correlation with the query terms. We use the Okapi BM25 term weighting scheme [8] to compute query-document similarity. The detail can be found in [5].

2.3 QUERY EXPANSION USING WIKIPEDIA

Wikipedia [3] is another source for our query expansion if the query terms have entries in Wikipedia. An Wikipedia entry article may have multiple titles. All these titles point to the same article. If this is the case, all the titles are considered as expanded terms. For example, when we search "*American Indian*", Wikipedia redirects the query to an entry titled "*Native Americans in the United States*". Then " *Native Americans in the United States*" is an expanded term. Accordingly, this entry article must also be highly related to the query. The important words from the article are also returned as expanded terms. The article consists of several sections including “summary”, “history”, “development”, “list of”, “references”, “see also”, “external links”, etc., in which the sections such as “see also”, “external links” are useless and thus are discarded. The terms, including concepts and individual content words, from the article are extracted. Each term is assigned a weight based on its frequency of occurrence in the article. The terms having the highest weights are considered as the expanded terms of the query.

2.4 QUERY EXPANSION USING WEB FEEDBACK

Using web feedback for query expansion is similar to the above local pseudo feedback. A query is submitted to a web search engine, which returns a ranked list of documents. In the top ranked documents, the terms that are highly correlated to the query terms are used for query expansion. We use Google in our system as the search engine.

3. RELAVENT DOCUMENT RETRIEVAL AND RANKING

After concepts identification and query expansion, an original query will be expanded with a list of concepts (if exists) and a list of expanded words. Every single term in the original query has weight of 1. The concepts are found from the original query. They also have weights of 1. The weights of the expanded terms are greater than 0 but smaller or equal to 1, where a weight of 1 means the expanded term is a synonym of the original query term. If an expanded term is returned from multiple expansion methods, the weights of this term from different methods are added together as the final weight for this term. If this final weight is greater than 1, it is set to 1. The idea is that the weight of an expanded term can not exceed that of an original term. The 20 top weighted expanded terms are chosen as the final expanded terms.

The similarity score between a query and a document consist two of parts: the concept similarity and the single term similarity (*concept-sim, term-sim*). The *concept-sim* is computed based on the identified concepts in common between the query and the document. The *term-sim* is the usual term similarity between the document and the query using the *Okapi* formula [8]. Each query term that appears in the document contributes to the term similarity, irrespective of whether it occurs in a concept or not. Since we emphasize that the concept is more important than individual terms, the *concept-sim* has a higher priority than the *term-sim*. Consider, for a given query, two documents d_1 and d_2 having similarities (x_1, y_1) and (x_2, y_2), respectively. d_1 will be ranked higher than $d2$ if either (1) $x_1 > x_2$, or (2) $x_1 = x_2$ and $y_1 > y_2$. Note that if $x_i>0$, then the individual terms which contribute to *concept-sim* will ensure that $y_i>0$. The calculation of *concept-sim* is described in [4].

4. SUBJECTIVE/OBJECTIVE SENTENCE CLASSIFICATION

After a ranked list of documents regarding a query is retrieved, the documents containing opinions toward that query need to be identified. We break a document to sentences and evaluate the subjectivity of each sentence. We say that a relevant document is an *opinioned relevant document* (ORD) if it is a relevant document from the retrieval process; and it contains opinioned sentences toward the query. The ORDs are our final objective for the Opinion Track. For each query topic, we collect a set of subjective sentences and a set of objective sentences respectively. Both sets must be related to the query topic. The subjective sentences are comments and reviews about the query. The objective sentences are descriptions of the query without any subjective opinions. A sentence classifier is then trained using these two training sets. The classifier can label a test sentence as either subjective or objective, which refer to opinioned and non-opinioned respectively. This classifier does not need to tell if the opinion in a subjective sentence is toward the query. Connecting an opinioned sentence to the query is another process after the sentence subjectivity classification is done, which will be described in Section 5. We want each query topic will have a customized classifier of itself because different queries have different language characteristics in their domains. For example, opinions toward movies may be "excellent performance", “funny”, etc. But they hardly appear in the opinions about a food such as Subway sandwich. So each query must have its own classifier.

4.1 USE WIKIPEDIA AND WEB TO GET OBJECTIVE SENTENCES

Given a query, we think that a dictionary entry of that query should be a high-quality data source of the objective text, since the dictionary definition should describe the term in an objective way. We use Wikipedia as the primary source to find the objective sentences for a query. Web search engine is used as a backup if Wikipedia does not have entry for a term. Given a query, we get the concepts in the query. These concepts are searched in Wikipedia to gather objective sentences. Once an entry is found for a concept, the whole entry contents are considered as objective data.

Case 1. The query contains one concept and Wikipedia returns one entry article for that concept. A query has one concept means that the whole query itself is a concept. The single Wikipedia article uniquely describes this concept. All the sentences in this article are saved as the objective sentences for this concept.

Case 2. The query contains one concepts and Wikipedia returns multiple articles for that concept. This means that the concept has multiple senses. We use Google to pick top 2 senses the returned articles. The concept, as a phrase, is submitted to Google. The search is restricted to be within the site of en.wikipedia.org by using “site:.en.wikipedia.org”. The documents returned from Google are all from Wikipedia. We save the sentences from the top 2 documents as the objective sentence set for that concept. We save the top 2 entries because we do the "title only" run in Blog Track. We do not know which entry to pick if a concept leads to multiple senses in Wikipedia. A safe move is to pick multiple entries in order to include the correct entry. But we do not want to keep all of the related entries neither, because too many articles introduce too many words, causing the real useful terms in the correct entry less significant, which in turn will decrease the quality of the sentence classifier to be generated. So we set 2 as the number of entries to save in this case. For example, a query "sun" has more than 40 related entries in Wikipedia about the star, companies, basketball teams, persons, newspapers, etc. Google ranked the entries of (1) “Sun as the star in our solar system” and (2) “the company Sun Microsystems” at the top 2 positions. So these two entry articles are saved.

Case 3. The query has one concept but Wikipedia does not have an entry for the concept. The concept as a phrase is submitted to Google without any search restriction to get a document list. The top n documents of this list form a set *A*. We also submit the concept as a phrase plus two additional words "blog", "comments" and a phrase “I think” to Google to get another document list, the top n documents of which form another set *B*. The sentences of the document set (A-B) are saved as the objective sentence set for the concept. We set n to 10 in our system. The rationale is that when Wikipedia does not have an entry for a concept, we search it via a web search engine. But the documents in the list B are very likely to contain opinions and thus should be excluded.

Case 4. The query contains multiple concepts. All the concepts that are not equal to the query itself form a partition of the query. Each of these concepts falls into one of the three cases above, and will get an objective sentence set respectively. All of these objective sentence sets are merged together as a single set for the whole query. For example, a query “*whale watching California*” contains two concepts: dictionary phrase “*whale watching*”, single term concept “*California*”, and a complex noun phrase “*whale watching California*”. The first two concepts partition the query. The last concept is the query itself and thus is not used in this case. Wikipedia returns one article for each of "*whale watching*" and "*California*" respectively. The sentences from these two articles are put together as the final objective sentence set for the query.

4.2 USE RATEITALL.COM AND WEB TO GET SUBJECTIVE SENTENCES

The comments and reviews toward a topic are mostly subjective contents, thus are good for the classifier training. In order to automatically collect high quality subjective data for various queries, we need a source, which covers a broad range of topics and has sufficient amount of subjective texts. Rateitall.com [10] is a web site asking users to provide opinions for various topics, including products, entertainment, people, politics, sports, foods, etc. The topics are organized in a hierarchical tree structure. We choose rateitall.com as the primary source of the comments and still use web search engine as a backup. We treat all the text of the comments from rateitall.com as subjective sentences. Although this is not true for every sentence, once we collect a large sentence set, the noise level of the set should be low.

The web site has a search interface to locate a topic directly. Each topic page has one short paragraph describing the topic, followed by a list of user reviews. Each review record contains comments. We utilize the search interface to find the topic page for a Blog Track query and extract the comments part. If the topic page only has small amount of comments, they may not be enough to train a high quality classifier later on. So, other than the searched topic page, we also collect comments from a certain number of the sibling topic pages of the searched page, in order to get enough subjective data. These sibling pages are the review pages of other topics, which are in the same category as the topic being searched. The language characteristics of these sibling pages and the searched page should be very similar. For example, the words in the comments toward "Chicago Bulls" should be very similar to those toward "Miami Heat" because they are both in the "NBA teams" category. A comment "good team" is suitable for both.

A Blog query is searched in rateitall.com. A list of relevant topics is returned. We pick the top 10 topics and compute a similarity score for each query-topic pair. The reason is that we think the original ranking from raiteitall.com does not always reflect the real importance of the topics. We need a re-rank process to adjust the ranking. A record in the search result list has a structure of <*rank, topic title, snippet, name of the parent directory, URL to the topic page, URL to the parent directory*>, where rank is a number between 1 and 10, the smaller the rank number, the higher the rank. The topic title is the name of the entity being commented, e.g. "*Chicago Bulls*". The snippet is a short paragraph containing the query terms from the description part of the topic page. We also follow the URL of the parent directory *p* to get the name of the parent directory of *p* (the name of the topic's grandparent node) for the re-ranking calculation.

Now we formally define the re-ranking function. Let $Q = \{q_1, q_2, \dots, q_n\}$ be a query where q_i (i = 1 .. n) is a query term. Let $L = <r_1, r_2, \dots, r_{10}>$ be the list of the top 10 search results from raiteitall.com, where r_i (i = 1 .. 10) = $< rank_i, title_i, snippet_i, parent_i, grandparent_i >$ is the *i_th* ranked record. The elements of r_i follow the descriptions in the last paragraph. Let s_{i_new} be the new ranking score of the record r_i. Then s_{i_new} is determined by a re-ranking function $h(r_i, L, Q)$. More in detail, $h(r_i, L, Q) = f(f_r(rank_i), f_t(title_i, Q), f_s(snippeti, Q), f_p(parent_i), f_g(grandparent_i))$

$$S_{i_{new}} = h(r_i, L, Q) = f\left[f_r(rank_i), f_t(title_i, Q), f_s(snippet_i, Q), f_p(parent_i, L), f_g(grandparent_i, L)\right]$$

The intuitions of these functions are: a higher original ranking score should contribute more to the new ranking score than a lower original ranking score does. This is to respect the ranking from rateitall.com. A result r_i has the query Q in its title, while another result r_j not, then $title_i$ should contribute more to the new ranking score than $title_j$. This means a title containing Q is more valuable than a title not containing Q. A result r_i has more query terms in its snippet than another result r_j, then the $snippet_i$ should contribute more to the new ranking score than $snippet_j$ does. The case of snippet is similar to that of the title. Given two parent names p1 and p2 found in L, p1 should contribute more to the new score than p2 does if the number of occurrence of p1 is larger than that of p2 in L. For example, 4 of the top 10 rateitall.com results for the query "*java*" come from the parent category of "*Computer & Technology Books*". This indicates that the query is more likely to be related to "*Computer & Technology Books*". Therefore, the four "*Computer & Technology Books*" related results should be considered more valuable than other results. The same idea applies to the grandparent names. Given two grandparent names g1 and g2 in L, g1 should contribute more to the new score than g2 does if the number of occurrence of g1 is larger than that of g2 in L,. According to these considerations, we designed the re-ranking function and the sub-functions as follows:

$$S_{rank} = 10 - \text{rank\_number} \qquad (1)$$

$$S_{title} = \begin{cases} 5, \text{if all query content words are found in the title} \\ 0, \text{otherwise} \end{cases} \qquad (2)$$

$$S_{snippet} = \text{instances of query content words found in the snippet} \qquad (3)$$

$$S_{parent} = \text{number\_of\_times\_of\_this\_category\_found\_in\_the\_top\_10\_results} \qquad (4)$$

$$S_{grandparent} = \text{number\_of\_times\_of\_this\_grandparent\_category\_appeared\_in\_the\_top\_10\_results} \qquad (5)$$

$$S_{re-rank} = S_{rank} + S_{title} + S_{snippet} + S_{parent} + 0.5 \times S_{grandparent}$$

In (5), we consider that the grandparent category have the same functionality as the parent category but less value. So we use a co-efficient (0.5) to adjust the score of (5) in the final formula.

Original	Re-ranked	Count
1	1	28
0	1	6
1	0	0
0	0	14
-	-	2

Table 1. Effect of re-ranking

Table 1 shows the effect of re-ranking. In the first and second column, 1 stands for the top ranked record is the best choice, 0 otherwise. A "hyphen" means no search result is returned. 6 of the 50 queries (second row) benefit from the re-ranking. Their top results in the re-ranked lists are better than the top ones in the original list. None of the queries gets worse top result after the re-ranking. After re-ranking, we choose the best result from the re-ranked result list according to the characteristics of the queries and the results.

Case 1. 1-word query Q; Q is found in the title of the top-ranked result of the re-ranked list; the top result is the only one having Q in the title in the list. In this case, the top-ranked result is chosen as the topic page for the query. This is a straightforward case.

Case 2. 1-word query Q; Q is found in the titles of multiple result records, or Q is found in the titles of non-top-ranked results of the re-ranked list. In this case, we build a vector of description terms for Q, and then compare it to the term vectors of the results. The result having the highest cosine similarity will be chosen as the answer. This description term vector consists of the 20 most frequent content words from the objective sentence of Q, which is generated in Section 4.1. Then for every result, the terms in the title, snippet, parent category name and grandparent category name form a term vector. The intuition is that the words in the result record can be considered as a short description of the corresponding topic page. The result that is most similar to the description vector should be chosen. For example, in the re-ranked list for the query "*sun*", all records have that term in the titles. The top ranked record is about a Jazz singer. Others are about NBA teams, songs, TV shows, star and company. The 20 most frequent stemmed terms from the objective sentence set are "*solar, earth, magnetic, energy, surface, corona, temperature, atmosphere, core, photosphere*". After computing the similarity between these 20 terms and those result records, the topic page in the "*Stars, Constellations, and Asterisms*" category is chosen.

Case 3. *1-word query Q; Q is NOT found in the title of any record in the re-ranked list.* The title can not give us any information in this case. We first find the parent category that has the largest number of records in the re-ranked list. Then we find the highest ranked record that has this category as its parent category. This record is chosen as the topic page for the query. The rationale is that none of the records is directly related to the query because Q is not shown in the titles. But rateitall.com does return results for this query. These results must be related the Q. The parent category is the possible common information among them. So we find the most frequent parent category the re-ranked list, and pick the highest ranked result from this category. For example, searching "PCI" returns 4 results, none of which has that term in the title. The top record is in "*International Politicians (Past & Present)*" category. Other three are in the *"Desktop computers"* category. So the top record from the "*Desktop computers*" category is chosen.

Case 4. Multi-word query Q; search the whole query as a PHRASE; all the query terms are found in the title of the top ranked result in the re-ranked list. We pick this top ranked result in the re-ranked list as the comment page for the query. Rateitall.com allow the user search several words together as a phrase, which means the words are in adjacent positions as they are submitted. If the phrase search returns result, and the top ranked result have all the query terms in title, this is a very strong indication that this top result should be the correct answer of the query. So we pick this top result. For example, searching "*blue book*" as a phrase gets five results. The top one has the exact phrase in title while others do not have both "*blue*" and "*book*" in titles. Thus the top one is chosen.

Case 5. Multi-word query Q; search the whole query as a PHRASE; no result is returned or not all of the query terms are found in the title of the top ranked result in the re-ranked list. This means the query, as a phrase, is too strong to search. We search Q again as a "bag of words" and results are returned. In this case, we examine the top 10 results in the re-ranked list, and pick the results that have different query terms in title and parent category. If such result records exist, we pick the highest ranked record from them as the comment page for the query Q. The rationale is that: Since the phrase search in the Case 5 does not return a desired result, the title does not hold all the query terms. We add the parent category as an additional place to search for the query terms. If part of the query is found in the title and another part of the query is found in the parent category, it is also a good indication of a good result. For example, searching "*hydrogen fuel automobiles*" as a phrase yields no result. While searching it as a bag of words, seven results are returned. One of the results has a title of "*hydrogen*" and a parent category of "*Alternative Fuels*". The term "*hydrogen*" is found in title and term "*fuel*" is found in the parent category name. The other result does not have any of the query terms in title. So this "*hydrogen*" topic page is picked for the query.

Case 6. Any other case that no result is returned. Rateitall.com does not have comments about the query. We submit the query Q plus words of "blog", "comments", and a phrase "I think" to Google; get the top 30 documents returned from Google; save the contents of these 30 documents as the subjective sentence set of query Q. The intuition is that the additional query words "blog", "comments" and "I think" all indicate existence of subjective comments. We search the query Q with these words, hoping to get documents containing comments. A query "*mutual fund predictors*" is such a case.

After we get a rateitall.com topic page for a query, all the comments in that page are saved. Furthermore, we randomly choose up to 59 (make the total number of pages no exceeding 60) sibling topic pages of the chosen page, save the comments in these pages too. The reason is that the comments from the best related page may not have enough content as the subjective training set for the sentence classifier, so we utilize the comments of the sibling pages in the same category to collect enough data.

4.2 TRAIN A SENTENCE CLASSIFIER BY USING SIGNIFICANT WORD FEATURES

The way we utilize the collected objective and subjective texts is to pick feature terms from the sets and use these feature terms to train the classifier. These features represent the language characteristics of the subjective and objective sentences about a query. We use word unigrams (single words) and bigrams (pair of words) as the feature types. We use the unigram to illustrate the feature term picking process. All distinct unigrams in a query's subjective and objective sentence sets are collected. For every distinct unigram *t*, the standard chi-square test [2] is conducted to test the hypothesis that t is distributed unevenly in the two sets. The term *t* is accepted as a feature term if the hypothesis holds.

	Have *t*	Does not have *t*	Row total
Subjective set	a_{11}	a_{12}	$a_{11}+a_{12}$
Objective set	a_{21}	a_{22}	$a_{21}+a_{22}$
Column total	$a_{11}+a_{21}$	$a_{12}+a_{22}$	Grand total $a_{11}+a_{12}+a_{21}+a_{22}$

Table 2. A chi-square test table.

In the test, four numbers related to *t* are counted: a_{11}: number of subjective sentences having *t*; a_{12}: number of subjective sentences not having *t*; a_{21}: number of objective sentences having *t*; a_{22}: number of objective sentences not having *t*. These 4 values are the observed values. For each of the observed value O_{ij}, the corresponding expected value E_{ij} is calculated as:

$$E_{ij} = \frac{row_{total_i} \times column_{total_j}}{grand_{total}}, \quad \text{i, j} \in \{1,2\}$$

The chi-square value of this observed value $(\text{Chi}^2)_{ij}$ is calculated as:

$$\chi^2(t)_{ij} = \frac{(O_{ij} - E_{ij})^2}{E_{ij}}, \quad \text{i, j} \in \{1,2\}$$

The final value for *t* is $\chi^2(t) = \sum_{i=1}^{2}\sum_{j=1}^{2}\chi^2(t)_{ij}$

The final chi square test value for *t* represents how different the subjective set and the objective set are in terms of *t*. The larger the value is, the more significant the difference is. A term with large chi-square value means its distributions in the two sentence sets are very different, thus this term is a good indicator of either the subjective sentence or the objective sentence but not both. After the chi square values of all the word unigrams are calculated, we rank these unigrams by descending order of their chi-square values. Unigrams with a chi-square value greater than a threshold are saved as feature unigrams. Other unigrams are discarded. We repeat the above procedure to process all the word bigrams to get the featured bigram set. We set the threshold to 5.02, which represents the significance level of 0.025.

When the featured unigrams and bigrams are ready, we convert all the subjective and objective sentences of the query to corresponding vector representations. Each sentence vector has a label indicating subjective or objective. All the feature unigrams and bigrams found in this sentence are listed in the vector. A term from the subjective (objective) data has a label of 1 (-1). We only record the existence of a featured term, but do not record its frequency in this sentence. Several opinion classification literatures [7] [9] have reported that using the existence information of a term instead of the frequency got higher accuracy. The Support Vector Machine (SVM) [6] with the default linear kernel is used to build the sentence classifier. All the subjective and objective sentence vectors are used as the training set of the SVM algorithm. The output is a SVM classifier. This classifier takes a sentence vector as the input, and outputs the predicted label (subjective or objective) and an associated score. Subjective sentence gets a positive score while objective sentence gets a negative score. The score represents the confidence level of the classifier to this answer. Larger absolute score value (toward infinity) means higher confidence, while a score close to 0 means low confidence.

5. OPINIONED DOCUMENT CLASSIFICATION

A ranked document set (about 1500 documents) regarding a query topic is obtained from Section 3. Each document in this set is decomposed into an ordered sentence list by a sentence splitter module. Each sentence is tested and labeled by the sentence classifier. If all the sentences are labeled as objective, the corresponding document is considered as not opinioned and is discarded. Otherwise, for each of the labeled subjective sentence, we get two sentences prior to it and two sentences following it. Then we search the original query terms and the expanded query terms within this five-sentence text window. There are three special cases:

(1) If the original query has one word and it is found in text window.
(2) If the original query has multiple words and at least two of the original query words are found in this text window.
(3) If (1) and (2) do not hold and at least three expanded terms are found in this window.

If any of these three cases happens, the subjective sentence is labeled as an *opinioned relevant sentence*. If none of the above three situations happens, the relevant sentence is considered as not relevant to the query and is discarded. A document having at least one opinioned relevant sentence is said to be an *opinioned relevant document* (ORD) of the query topic.

Two ranking methods are tried to rank the ORDs. The first method is to keep the original ranking order of the relevant documents from the information retrieval step in Section 3. Then remove all the documents that are not opinioned. The ranking formula is:

$$Score_{rank}(D,Q) = Sim(D,Q) \times I(D,Q) \qquad (1)$$

where
Sim(D, Q) is the relevance score from the retrieval step.

I(D, Q) = 1 if document D contains relevant opinion sentence about the query Q, 0 otherwise.

In Section 4.2, the SVM classifier gives every tested sentence a classification score representing how subjective or objective the sentence is. An ORD must contain one or more opinioned relevant sentences. We use the sum of the classification scores of the opinioned relevant sentences in an ORD as the second ranking method. The ranking formula is:

$$Score_{rank}(D,Q) = \sum_{s \in \{opinioned\ relevant\ sentences\ in\ D\}} \left[Score_{classification}(s) \times relevant(s,Q) \right] \qquad (2)$$

where

$score_{classification}(s)$ is the score given by the SVM classifier to a sentence *s*.

relevant(s, Q) = 1 if the sentence *s* satisfies any of the three cases described above, 0 otherwise.

Given a query, we rank all the documents in descending order of the scores calculated by (1) and (2) respectively. Two ranking results are submitted as two runs. The "uicsr" run uses formula (1) and "uicst" run uses the formula (2). Both runs are title-only automatic runs. The overall scores of the two runs are listed in Table 3.

RUN	MAP	GMAP	R-Precision	P@10
UICSR	0.1636	0.0921	0.2522	0.4380
UICST	0.1885	0.1083	0.2771	0.5120

Table 3. Summary of UIC Blog Track Title-Only Automatic Runs

6. CONCLUSIONS

In the opinion task of the TREC 2006 Blog Track, we develop a two-step model to retrieve documents that have opinioned content about a query topic. We adopt the concept identification and multi-source query expansion techniques in the relevant document retrieval step. Subjective/Objective sentence classification is used in the opinion identification step. Two ranking methods are used to create ranked opinioned relevant document set to answer a query.

REFERENCES

[1] Ricardo Baeza-Yates, Berthier Ribeiro-Neto: *Modern Information Retrieval*, Addison-Wesley, 1999.

[2] Chernoff H, Lehmann E.L. The use of maximum likelihood estimates in $\chi 2$ tests for goodness-of-fit. The Annals of Mathematical Statistics 1954; 25:579-586.

[3] en.wikipedia.org

[4] Shuang Liu, Fang Liu, Clement Yu, and Weiyi Meng. An Effective Approach to Document Retrieval via Utilizing WordNet and Recognizing Phrases. In Proceedings of the 27th Annual International ACM SIGIR Conference. 2004.

[5] Shuang liu and Clement Yu. UIC at TREC 2005: Robust Track. In Proceedings of TREC Conference. 2005.

[6] T. Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Proceedings of the European Conference on Machine Learning (ECML), Springer, 1998.

[7] Rahman Mukras and John Carroll. A Comparison of Machine Learning Techniques Applied to Sentiment Classification. 2004.

[8] S. Robertson, S. Walker Okapi/Keenbow at TREC-8, 1999.

[9] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). 2002.

[10] www.rateitall.com

[11] Wei Zhang, Shuang Liu, Chaojing Sun, Fang Liu, Weiyi Meng, and Clement Yu. Recognition and Classification of Noun Phrases in Queries for Effective Retrieval. A Technical Report. Computer Science Department, Univ. Of Illinois at Chicago. 2006.