Opinion Retrieval from Blogs

Wei Zhang
Department of Computer Science
University of Illinois at Chicago
851 S Morgan St
Chicago, IL 60607, USA
wzhang@cs.uic.edu

Clement Yu
Department of Computer Science
University of Illinois at Chicago
851 S Morgan St
Chicago, IL 60607, USA
yu@cs.uic.edu

Weiyi Meng
Department of Computer Science
Binghamton University
Vestal Parkway East
Binghamton, NY 13902, USA
meng@cs.binghamton.edu

ABSTRACT
Opinion retrieval is a document retrieval process, which requires documents to be retrieved and ranked according to their opinions about a query topic. A relevant document must satisfy two criteria: relevant to the query topic, and contains opinions about the query, no matter if they are positive or negative. In this paper, we describe an opinion retrieval algorithm. It has a traditional information retrieval (IR) component to find topic relevant documents from a document set, an opinion classification component to find documents having opinions from the results of the IR step, and a component to rank the documents based on their relevance to the query, and their degrees of having opinions about the query. We implemented the algorithm as a working system and tested it using TREC 2006 Blog Track data in automatic title-only runs. Our result showed 28% to 32% improvements in MAP score over the best automatic runs in this 2006 track. Our result is also 13% higher than a state-of-art opinion retrieval system, which is tested on the same data set.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *retrieval models, selection process*. I.2.7 [Artificial Intelligence]: Natural Language Processing – *text analysis*.

General Terms
Algorithms, Performance, Experimentation.

Keywords
Opinion Retrieval, Blog Retrieval, Information Retrieval, TREC.

1. INTRODUCTION
The fast growth of blog (derived from "Web Blog") web sites has created a highly active portion of the World Wide Web. People publish blogs, a kind of journal style web pages, which provide news, comments, and personal diary-like articles. Readers can leave feedbacks to the blogs. Major web search engines such as Google [7] have launched blog search services. But these blog search tools are just the applications of the traditional web search techniques in the blog domain, in that, given a query, they only search for fact-related information in the blogs. However, the opinions about various topics, which are very important feature of the blog documents, are not searched by these tools. In 2006, the Text REtrieval Conference (TREC) brought up a "Blog Track" for the first time focusing on information retrieval in the blog documents [17]. The major task of this track was the "opinion retrieval". An opinion retrieval system is required to locate blog documents expressing opinions about a query in a large blog collection. Identification of the polarity of the query-relevant opinions in the documents is not required. A relevant document should satisfy two criteria: relevant to the query, and contains opinions or comments about the query, no matter if they are positive or negative. For example, given a query "Audi", a relevant document must contain opinions about either Audi cars or the company itself. A document telling a driver's feelings about his Audi A4 is relevant in opinion retrieval. However, a document only describing mechanic features of an Audi A4 is not relevant. Although the machine learning and natural language processing communities have worked on finding opinions about targets such as products [15], movie reviews [18] in documents, they usually assume that the documents have already contained the relevant opinions. They do not address the general problem of how to retrieve the opinionative documents from a collection. On the other hand, the traditional information retrieval (IR) community only focuses on the fact-oriented retrieval. It does not handle the subjective contents about the queries in the documents.

In this paper, we present an opinion retrieval algorithm that retrieves blog documents according to the opinions and comments about a query that the blog documents contain. We first define some concepts for the opinion retrieval.

- A *query topic* (*query* for short) is a list of words to be searched in a document collection. The queries are the inputs to an information retrieval system.
- The *subjective texts* are the *comments* or *opinions* about a target in a document. In a blog document, the comments can come from the original post of the blog, or from the feedbacks added after the original post by other people. In our paper, "subjective texts", "subjective contents" of a document, "comments" and "opinions" all have the same meaning.
- The *relevant opinions* of a query are the comments or opinions that are about that query. For instance, in the *Audi* example introduced above, opinions about the Audi cars or

CIKM'07, November 6–8, 2007, Lisboa, Portugal.

company are relevant opinions. Opinions about a gas station in the same document are not relevant opinions.

- The *opinionative documents* are the documents containing any types of opinions, which are not necessarily related to a query.
- The *topic relevant documents* (relevant documents for short) of a query are the ideal output of a document retrieval system. Such topic relevant documents may or may not contain opinions about the query.
- The *Relevant Opinionative Documents* (*ROD*) of a query are documents that contain *relevant opinions* about the query. A list of ROD is the ideal output of an opinion retrieval system.

We consider the ROD as a subset of the intersection of the *topic relevant documents* and the *opinionative documents*. Accordingly, our opinion retrieval algorithm has three components. (1) An IR component retrieving the topic relevant documents, no matter if they contain opinions or not. We utilize an IR system built for a past TREC task for this purpose. (2) An opinion identification component that retrieves the opinionative documents. We build support vector machine (SVM) classifiers to identify the opinionative documents. Two types of SVM classifiers are constructed. One is query-dependent and the other is query-independent. (3) A ranking component. It has a NEAR operator to find the ROD by looking for the query-relevant opinions. Then the ranking component utilizes multiple strategies to rank the retrieved ROD by using the query-document similarities and the intensities of the query-relevant opinions in the documents. Our algorithm is tested on the TREC 2006 Blog Track data [17]. Our contributions are:

1. Present a system for the new research field of opinion retrieval.
2. Study the effects of the variations of the components in our system on retrieval effectiveness.
3. Study the effects of having different sets of opinion features on retrieval effectiveness.
4. Experiments show that the retrieval effectiveness of our system has significant improvements over the best results published in the recent TREC 2006 Blog Track by 28% to 32%. Our result is also 13% better than a state-of-art opinion retrieval system tested on the same data set.

The rest of this paper is organized as follows. Section 2 reviews the related works. Section 3 describes the IR component of our algorithm. Section 4 describes the SVM opinion finding component. Section 5 describes method of finding query-relevant opinions. Three document ranking strategies are also introduced in Section 5. Experimental results are reported in Section 6. Conclusions are given in Section 7.

2. RELATED WORKS

The opinion retrieval was the major task in the new TREC 2006 Blog Track. 14 groups participated in this task. The groups with the top performances were [29][31][32][33].

Mishne [29] adopts fact-oriented IR, dictionary-based opinion expression detection and spam filtering as three major components in his system. This system not only utilizes the blog documents being tested, but also the RSS seeds as additional training data. Mishne [30] recently reported an opinion retrieval system that utilizes the publishing dates of the documents as a feature in the retrieval. This system also compares the contents of actual blog posts and the RSS documents to calculate the proportion of the comments in a blog document. Query-dependent spam filters, which are in the form of support vector machine, are used to further remove spam documents. This system achieved a MAP score of 0.2411, which was higher than the best score in TREC 2006 Blog Track. In the experiment part, we show that the score of our system is about 13% higher than that of this system.

Zhang and Yu [31] adopt concept-based IR [16], machine-learning approach based opinion detection and two separate opinion similarity functions in their system. Their system, which was an earlier version of our system, obtained the best performance among the groups using the title-only queries [17].

Yang et al. [32] adopt IR components that utilizes proximity match and phrase match. Their opinion component adopts frequency-based heuristics, special pronoun patterns and adjective/adverb-based heuristics. Although this system reported the best performance among all the participating groups, it has a tuning module involving human-feedback, which makes it is not a purely automatic system. Furthermore, their best run was achieved by using the title-description-narrative queries, which provide more information than title-only queries [17].

Oard et al. [33] adopt passage retrieval; both dictionary-based and machine-learning based sentiment term selection methods; a passage sentiment score is calculated by using all the sentiment terms in that passage.

Other related works come from the machine learning and the natural language processing communities, focusing on identifying the subjective texts in the documents and classifying the subjective/objective nature of such texts.

The subjectivity orientations of words or phrases have been studied. Hatzivassiloglou and McKeown [5] studied the adjectives as evidence of subjective texts. Hatzivassiloglou and Wiebe [6] investigated sentence subjectivity classification. A method is proposed to find adjectives that are indicative of positive or negative opinions. Wiebe et al. [26] identified subjective language features, such as low-frequency words, word collocations, adjectives and verbs, from corpora and used them in the sentiment classification. Esuli and Sebastiani [4] classified the orientation of a term based on its dictionary glosses. Whitelaw, Garg, and Argamon [25] used short phrases called “appraisal groups” subjective language features for sentiment classification.

In the opinion identification works, Liu, Hu and Chen [15] and Popescu and Etzioni [20] both extracted product features and their opinions from the Web. They use heuristic rules and supervised learning techniques to find product features and opinions. Recently, Jindal and Liu [10] also studied the opinions in comparative sentences using the support vector machine and the naïve Bayes classifiers with both manual and automatic rules. Other opinion identification works include [12][28][9].

In the document-level opinion classification, Das and Chen [2] use a manually crafted lexicon in conjunction with several scoring methods to extract investor sentiment from stock message boards. Tong [22] generates sentiment (positive and negative) timelines by tracking online discussions about movies over time. Pang et al.

[18] examines several supervised machine-learning methods for positive/negative sentiment classification of movie reviews. Pang and Lee [19] also studied the problems of multi-category classification of movie reviews by using supervised learning.

Various types of machine-learning techniques have been applied to the opinion classification task. Turney [23] applies an unsupervised learning technique based on mutual information between document phrases and the words "excellent" and "poor" to find indicative words of opinions for classification. Dave et al. [3] also experiment with a number of learning methods for review classification. They show that the classifiers perform well on whole reviews, but poorly on sentences because a sentence contains much less information.

Ku et al. [14] summarize blog documents by their major topics and the related opinions.

3. TOPIC RETRIEVAL

The IR component of our opinion retrieval algorithm is a traditional document retrieval procedure. It retrieves all of the query-relevant documents from a document collection. The idea is that, given a query, the relevant opinionative document (ROD) set is a subset of the query topic relevant document set. An ideal IR step should retrieve all the topic relevant documents; all of the ROD should be retrieved accordingly. The subjectivities of the documents are not the concern of this step, as they will be identified in the classification module. Theoretically, any document retrieval system fits here, such as the Okapi formula based systems or the systems based on various language models. In our implementation, we adopt a system derived from [16], which has reported the best GMAP (Geometric Mean Average Precision, the official measure used for TREC 2005 Robust Track) score using TREC 2005 Robust Track data. It has the characteristics of concept recognition and query expansion. It places more emphasis on concept matching than individual term matching. We only describe the key points of this system. The details can be found in [16].

3.1 Concept Identification

A concept in a query is defined as either a multi-word phrase consisting of adjacent query words, or a single word that does not belong to any other concept [34]. Concept identification is a query pre-processing procedure, which partitions an original query to these concepts. Concept is introduced to improve retrieval effectiveness. For example, the words in a query "*Happy Feet*" (a movie title) can be recognized as a phrase. The query should be searched as a single phrase rather than as two individual words. We designed a phrase recognition algorithm, which involves dictionaries, statistics from document sets, Web and natural language processing tools, and integrate it into our opinion retrieval algorithm. The detail of this phrase recognition algorithm can be found in [34]

3.2 Query Expansion

We adopt three query expansion methods. One method is a dictionary-based method that utilizes Wikipedia [20] to find an entry page for a phrase or a single term in a query. If it has such an entry, all the titles of the entry page are expanded as synonyms of the query concept. For example, if we search "PC game" in Wikipedia, it is re-directed to an entry page titled "personal computer game", which is then added as a synonym of "PC game" in the query. The synonyms are treated the same as the original query terms in the retrieval process. The content words in the entry page are ranked by their frequencies in the page. The top k terms are returned as potential expanded terms.

We adopt the local context analysis [35] as the second query expansion method. Our method combines the global analysis and local feedback. The query in the form of concepts is used to retrieve a ranked document list from the given document collection. The terms in the top n (a threshold) documents are ranked according to the formula in [35]. The top k terms are returned as the potential expanded terms of this method.

The last query expansion method is a web-based method. It is similar to the local pseudo feedback expansion but using web documents as the document collection. The query is submitted to a web search engine, such as Google, which returns a ranked list of documents. In the top m documents, the top k terms that are highly correlated to the query terms are returned.

Each expansion method generates terms with weights to be added to the original query. All the expanded terms of all the concepts of a query are put together. If an expanded term is returned by more than one method, the sum of its weights from the different methods is its weight. The weights of the expanded terms are normalized to values in (0, 0.7). The idea is that the original query terms and concepts all have weights of 1, while the expanded terms should have lower weights. The 20 top weighted expanded terms are chosen as the final expanded terms for each query.

3.3 Relevant Document Retrieval and Ranking

After the recognized phrases and the expanded terms are added to the queries, the queries are sent to document retrieval system to retrieve relevant documents. We adopt the IR system by Liu et al. [16]. It allows both phrases and single terms to be in the query. Let d be a document, Q be a query, C be the set of phrases of Q that are from the phrase recognition, E be the set of expanded terms of Q that are obtained from query expansion, W be the set of corresponding term weights of the terms in E, calculated in Section 3.2, *idf(p)* be the inverse document frequency weight of a concept p, *Okapi(t, d)* be the Okapi/BM-25 score [21] of a term t in d, the similarity between the query Q and a document d, *Sim(d, Q)*, is defined as a pair of (phrase-similarity *Sim_p(d, Q)*, term-similarity *Sim_t(d, Q)*):

$$Sim(d,Q) = \left(Sim\_p(d,Q), Sim\_t(d,Q)\right) \quad (3.1)$$

where

$$Sim\_p(d,Q) = \sum_{p \in C} idf(p)$$

$$Sim\_t(d,Q) = \sum_{q \in Q} Okapi(q,d) + \sum_{t \in E} W(t) \times Okapi(t,d)$$

The phrase-similarity is defined as the sum of the idf (inverse document frequency) weights of the phrases in common between the document and the query. If a document does not have the recognized phrase, the *Sim_p* is 0. The term-similarity is the usual term similarity between the query and the document, which is computed by using Okapi formula. Each query term that appears in the document contributes to the term-similarity, no matter it is in a query phrase or not. The *Sim_p* has a higher priority than *Sim_t*. The retrieved documents of a query are ranked in

descending order of their *Sim_p* values. When documents have the identical phrase similarity value, they are ranked in descending order of their term similarities. So given a query, two documents D1 and D2 have similarities (x1, y1) and (x2, y2), respectively. D1 will be ranked higher than D2 if (1) x1>x2, or (2) x1=x2 and y1>y2.

4. OPINIONATIVE DOCUMENT RETRIEVAL

This component is to identify the opinionative documents in a document set, regardless of what the opinions are about. The documents retrieved from topic retrieval can be separated into three sets: (1) no opinion, (2) opinionative but not relevant to the query, and (3) opinionative and relevant to the query (ROD). In this step, we distinguish the documents of (2) and (3) from those in (1). A statistical classifier is built to label the opinionative texts in a document. The document is decomposed into sentences. The classifier labels each of them as either subjective (opinionative) or objective (non-opinionative). The document is opinionative if at least one of its sentences is labeled as subjective by the classifier. We adopt a support vector machine (SVM) classifier that uses unigrams (single words) and bigrams (two adjacent words) as features. The vectors are presented in a presence-of-feature form, i.e. only the presence or absence of each feature is recorded in the vector. The number of occurrences of the feature is not recorded. This classifier-feature setup had been shown to be among the best configurations by Pang [18]. We use the SVM-Light [10] with its default setting, which is the linear kernel function, as the SVM implementation.

4.1 Collecting Data for Classifier Training

Given a query, we collect a subjective document set and an objective document set related to that query as the training data of the classifier. We expand the idea in [18], which only uses movie reviews as subjective texts and movie story narratives as objective texts: Query-dependent documents are collected for each query separately. The subjective documents of a query are gathered from reviews web sites such as rateitall.com and epinions.com. We use the concepts and single terms obtained in Section 3.1. For each such single term or concept, we search its related reviews via the review site's search interface. There are two possible cases:

Case 1. There is only one group of reviews available for this term/concept. Download the reviews in this group. Set the downloaded documents as the reviews of this term/concept. This refers to the case that the term/concept has only one sense. No sense disambiguation is needed.

Case 2. There are more than one group of reviews available. It means that the query term/concept may have multiple senses. For example, "*Chicago*" can refer to a city, a music band, or a movie. We pick one review group by sense disambiguation. A word vector is formed using other original query terms and the expanded query terms to represent the meaning of the term/concept being searched. The descriptions of the review groups are available from the web site. The words in each description form a vector as the sense for the corresponding review group. Cosine similarities are calculated between the query sense vector and all the review sense vectors. The review group having the highest similarity score is downloaded as the subjective documents for the query term/concept.

After we get the review documents for all the concepts of a query, these documents are put together to form a single document set for the whole query. We assume that the contents of these review documents are all subjective. Although this is not 100% true, as we collect a large amount of reviews, the subjective portion should be dominant so that the effect of the objective portion can be neglected. In order to get large amount of data, we also download additional reviews from the original reviews' siblings in the review web sites. The siblings refer to the entities in the same category as a query term/concept. For example, if a query is "*Toyota Camry*", in rateitall.com, reviews about "*Ford Fusion*" are also collected because they are both in the "*Sedan*" category. These sibling reviews are extracted by using the web site navigation tree among the entities: after the entry for a query term/concept is located and its reviews are downloaded, go to the parent node of this entity, get other child nodes of this parent, and download their reviews. For each query, we collect reviews from up to 60 of its siblings. Another way of collecting the subjective documents is to submit the original query plus some "opinion indicator phrases" such as "*blogs OR reviews OR comments*", "*I think*", "*I don't think*", "*I like*" or "*I don't like*" to a search engine and get the returned documents. The intuition is that these phrases are indicative of opinions. If the documents contain both the query and one of these phrases, it is very likely that they contain opinions relevant to the query. So we assume that the contents of these documents are subjective too. Google is used as the search engine in our implementation. For each of the "query + indicator phrase" combination, we take the top 3 returned documents. The documents from the review web sites and the search engine form the subjective document set of a query.

The objective training documents are collected from Wikipedia in a similar way. The dictionary entry pages are high-quality objective data source, since the definitions describe things without emotion. We submit each of the concepts and non-concept single terms of a query to Wikipedia to get the corresponding definition pages. All the contents of these pages are assumed to be objective. In order to have enough statistics, we also submit the original query, with *a restriction of not containing any terms "reviews", "comments", "opinion", "posts"*, to a web search engine. The intuition of this is that a document is likely to be non-opinionative if none of these words are found. The top 2 documents returned from Google are collected. The documents from Wikipedia and Google form the objective document set of a query.

4.2 Selecting Useful Features

We do not use all of the unigrams and bigrams in the collected documents as the features of the SVM classifier. Only a subset of them is chosen via the Pearsons chi-square test[1]. Yang [27] reported that chi-square test was an effective feature selection approach. To find out how dependent a feature *f* is with respect to the subjective set or the objective set, we set up a null hypothesis that *f* is independent of the two categories with respect to its occurrences in the two sets. A Pearsons chi-square test compares observed frequencies of *f* to its expected frequencies to test this hypothesis by a contingency table. Table 1 shows the contents of a contingency table. To calculate the chi-square value of a feature *f*, all the subjective and objective documents are decomposed to sentences. All sentences in the subjective documents are labeled as subjective. All those in the objective documents are labeled as objective. The observation O*ij* in Table 1 is counted as the

number of sentences having/not-having *f* in the subjective/objective set respectively. For example, O_{12} is the number of sentences not having *f* in the subjective sentence set.

Table 1. Contingency table for Pearsons chi-square

	F	┐*f*	Row total
Sub. set	O_{11}	O_{12}	$O_{11}+O_{12}$
Obj. set	O_{21}	O_{22}	$O_{21}+O_{22}$
Col. total	$O_{11}+O_{21}$	$O_{12}+O_{22}$	Grand total $O_{11}+O_{12}+O_{21}+O_{22}$

The independence of *f* is tested by calculating its chi-square value

$$\chi^2(f) = \sum_{i=1,2} \sum_{j=1,2} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

where E_{ij} is the expected frequency of case *ij* calculated by

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

The larger the chi-square value, the more class-dependent *f* is with respect to the subjective set or the objective set. We keep a feature if it has a chi-square value no less than 5.02, which corresponds to the significance level of 0.025. We assume that it is class dependent, and thus is used as a feature in the SVM classifier. During the text processing, we only do Porter stemming but do not remove any stop words. After the features are selected, all the subjective and objective sentences are represented as feature-presence vectors, where the presence or absence of each feature is recorded [18]. These vectors are used to train the SVM classifier. The classifier is generated in a *one-classifier-per-query* way, as the subjective and objective documents are collected according to the query. So each query has its own features and its own customized classifier, because we think that the features could be query dependent. For example, "*delicious*" may appear in opinions about foods, but hardly appear in opinions about a car.

4.3 Building the SVM Opinion Classifiers

4.3.1 Query-Dependent Classifier

Since people tend to use different words in different domains, for example, "*tasty*" in opinions about foods but hardly in opinions for automobile, and "*dependable*" vice versa, we want to build the query-dependent classifiers at first. It means that given a set of *n* queries, each query gets its subjective and objective training data sets in section 4.1, n queries result in *n* subjective/objective training sets. Each of these n training sets is used to select features and build a classifier only for the corresponding query. This results in *n* classifiers.

4.3.2 Query-Independent Classifier

A query-dependent (Q-D) classifier might be accurate but its efficiency could be a problem, if applied in practice. We construct another type of classifier, namely query-independent (Q-I) classifier. It is built as follows: given the same query set and the training data sets of section 4.3.1, the training data from all queries is used to build a classifier, so that the classifier is independent of the actual query. We will compare the performance of Q-D and Q-I classifiers via experiments.

4.4 Applying the SVM Opinion Classifiers

When the SVM opinion classifier for a query is ready, it tests the documents that are retrieved by the IR component of our system in response to the query. Such a document is decomposed into sentences. The classifier takes one sentence each time, outputs a subjective label and a numeric score for this sentence. The score represents the classifier's confidence to its decision of the label. The score is positive if the classifier considers the sentence to be subjective, or negative if it considers the sentence to be objective. A retrieved document is labeled as opinionative if it contains at least one subjective sentence labeled by the classifier.

5. FINDING THE RELEVANT OPINIONATIVE DOCUMENTS

5.1 Finding Query Relevant Opinions

The SVM classifier only determines if sentences in a documents are opinionative or not, but does not know if the opinions are about the query or not. We use a *NEAR* operator to classify an opinionative sentence as either relevant or not relevant to the query. This *NEAR* operator uses a text window to cover five sentences: a subjective sentence labeled by the classifier, the two adjacent sentences preceding it and the two after it. Then it searches for the query terms in this text window:

(1) Search for at least two words of the original query words or their synonyms in the text window, if the original query contains at least one multi-word phrase. The two terms do not have to be in the same phrase, because the document has been retrieved by the query, and some references to the query topic might be abbreviated. For example, a query "*California whale watching*" contains a phrase "*whale watching*". A sentence "I really enjoyed following the *whales* when I was in *California*" contains opinions about the query. It contains two query terms but not the phrase.

(2) Search for the original query word or at least one of its synonyms if the original query contains only one word.

(3) Search for at least one of the original query words or their synonyms, and at least one expanded query words, so that totally at least three words are found, if it is a multi-word query but not containing any multi-word phrase.

The *NEAR* operator outputs *true* if any of these three cases happens in the text window or *false* otherwise. We call the subjective sentence in that text window a *relevant opinionative sentence* (ROS) if NEAR outputs true. A document is considered as a ROD of the query if it contains ROS. The intuition is that people express opinions either directly or indirectly to a target. They can express the opinion in the same sentence with the target, or in a describe-and-then-comment way. We assume that this *NEAR* operator covers various situations. In summary, a document *d* is classified as a ROD of a query *Q* if (1) at least one of *d*'s sentences is classified as a subjective sentence by the SVM classifier of *Q*, and (2) at least one of *d*'s subjective sentences is a ROS recognized by the NEAR operator. If *d* is identified as a ROD, the sum of the classification scores of its ROS, which are given by the SVM classifier, is set as *d*'s opinion similarity score about the query.

5.2 Relevant Opinionative Documents Ranking

We have identified some documents as the potential ROD of a query. Now we present three opinion retrieval similarity functions to rank these documents.

5.2.1 Rank by Topic Retrieval Similarity Score

The first similarity function simply uses the similarity scores from the document topic retrieval step, as the opinion retrieval similarity scores. The idea is to test if the opinion retrieval similarity has a relationship with the intensity of the query terms and the expanded query terms in the documents. Let *d* be a document. Let *D* be the document set that are classified as ROD of a query *Q*, by the *NEAR* operator. Let *Sim(d, Q)* be the document topic retrieval similarity function defined in formula 3.1, this opinion retrieval similarity function is defined as

$$OSim_{ir}(d,Q) = \begin{cases} Sim(d,Q), & \text{if } d \in D \\ 0, & \text{otherwise} \end{cases} \quad (5.1)$$

We call formula 5.1 the "*ir similarity function*" later on. When it is applied, the documents classified as ROD are ranked in descending order of $OSim_{ir}$ scores.

5.2.2 Rank by Document Opinion Similarity Score

In Section 5.1 we set the sum of the classification scores of the ROS of a document *d* as *d*'s opinion similarity score. We use this score as our second function. Let *d* be a document. Let *D* be the set of documents that are classified as ROD with respect to a query *Q* by the *NEAR* operator. Let *R(d)* denote the set of relevant opinionative sentences in a document *d* in *D*. Let *s* be a sentence, and *score(s)* be the classification score of *s* given by the SVM classifier. This similarity function is defined as

$$OSim_{stcs}(d,Q) = \begin{cases} \sum_{s \in R(d)} score(s), & \text{if } d \in D \\ 0, & \text{otherwise} \end{cases} \quad (5.2)$$

This function takes the scores of the relevant opinionative sentences into consideration. We call it the "*stcs similarity function*" where *stcs* stands for "sentence score". When it is applied, the documents classified as ROD are ranked in descending order of $OSim_{stcs}$ scores.

5.2.3 Rank by Relevant Opinionative Sentence Count

Formula 5.2 uses the detailed classification scores of all the relevant opinionative sentences. We think the size of the ROS set could be another measure. Intuitively, the size of the ROS set should also reflect the intensity of the relevant opinions in a document. Let *R(d)* be the same *R(d)* as defined in Section 5.2.2. The third similarity function is defined as

$$OSim_{stcc}(d,Q) = \begin{cases} |R(d)|, & \text{if } d \in D \\ 0, & \text{otherwise} \end{cases} \quad (5.3)$$

We call this the "*stcc similarity function*" where "*stcc*" stands for "sentence count". When it is applied, the documents classified as ROD are ranked in descending order of $OSim_{stcc}$ scores.

5.2.4 Combine the Similarity Functions

Formula 5.1 only utilizes the document retrieval score, while formula 5.2 and 5.3 only use opinion related information. We want to see if the combination of these two types of scores can improve the opinion retrieval performance. So we create the combined similarity functions in a general form of

$$OSim_{ir\_stcs} = a \times OSim_{ir} + (1-a) \times OSim_{stcs} \quad (5.4)$$

$$OSim_{ir\_stcc} = b \times OSim_{ir} + (1-b) \times OSim_{stcc} \quad (5.5)$$

In (5.4) and (5.5), all the $OSim_{ir}$, $OSim_{stcs}$, and $OSim_{stcc}$ scores have been normalized to be within the range of [0, 1]. Various values of *a* and *b* between 0 and 1 will be tested, so that we can have an empirical understanding of the optimization of these two coefficients.

6. EXPERIMENTAL RESULTS

We have implemented our algorithm as an operational system. It is tested on the TREC 2006 Blog Track query and document set [17]. The query set contains 50 queries. They cover domains of movies, TV programs, politicians, politics, sports, electronic products, entertainment, health and food, celebrities, and organizations. Each query has a title, a description and a narrative. A query title resembles a typical query submitted to a web search engine. Among the 50 query titles, there are 16 one-term queries, 23 two-term queries, 8 three-term queries, 2 four-term queries and 1 five-term query. The description is a sentence explaining the meaning of the query in more detail. The narrative is a short paragraph about what types of documents are relevant and what are not. The document collection contains over 3.2 million blog documents (88.8GB) collected from various blog web sites between December 2005 and February 2006. The Track also releases the official golden standard for each query. So the real ROD set for each query is given. A retrieval system may return at most 1000 ranked documents for each query from the blog document set. Although our system is able to identify the opinions relevant to a query at the sentence level, it needs to follow the TREC requirements to rank and return whole documents as output. We conduct the *automatic, title-only* experiments, which use only the query titles as the original queries, and have no human interaction involved. The experiments are designed so that the performances of various parts of our algorithm are tested and analyzed. The experimental results are evaluated by the golden standard, and presented in Mean Average Precision (MAP) scores.

6.1 Performances of Individual Components

Our algorithm contains three components: IR, opinion classification and similarity ranking. We first conduct experiments testing how these components perform. We set seven system configurations where configuration (3), (4) and (5) are the full configurations using different opinion similarity functions:

(1) **IR**: a system only utilizes the IR component. The retrieved documents from the IR component are considered as the RODs, and are compared to the golden standard. This is to test the relationship between topic relevance and the opinion relevance: if higher topic relevance results in higher opinion relevance in a blog document collection.

(2) **IR+OC**: IR component plus the opinion classification module. For each query, the corresponding query-dependent opinion classifier checks the retrieved documents from IR component. All the subjective sentences labeled by the opinion classifier are considered as query-relevant. The documents labeled as subjective are considered as the RODs.

This is to test if detecting any kind of opinions can improve the opinion retrieval effectiveness.

(3) **IR+OC+NEAR**: configuration (2) plus the NEAR operator of the ranking component. This is actually the full configuration of our algorithm when using the $OSim_{ir}(d,Q)$ similarity function (formula 5.1), since *ir* function does not re-rank the documents that pass the OC and NEAR filters. This is to test how the distance between query terms and opinionative terms in the same document, as a filter, affects the retrieval effectiveness.

(4) **IR+OC+NEAR+STCS**: configuration (3) plus the *stcs* opinion similarity function. This is to test if ranking the retrieved documents by the density of their relevant opinions can improve the retrieval effectiveness.

(5) **IR+OC+NEAR+STCC**: Identical to (4) but *stcs* is replaced by *stcc* function.

(6) **IR+OC+STCS**: Configuration (2) plus the *stcs* opinion similarity function. This is to test how the missing of NEAR operation affects the similarity function.

(7) **IR+OC+STCC**: Identical to (6) but *stcs* is replaced by *stcc* function.

In these configurations, the opinion classification module is built as follows: the subjective documents for SVM training are from rateitall.com. Google is used to obtain subjective documents only when no subjective documents are obtained from rateitall.com. In addition, only the first of the 5 "subjective indicator phrases" in Section 4.1 is used (Section 6.2 will use all these 5 phrases). The objective documents are from Wikipedia. Google is used to obtain objective documents only when no objective documents are obtained from Wikipedia. Each query-dependent SVM opinion classifier has 2,661 features on the average. All the above configurations are tested by using the 50 queries in automatic, title-only runs. The overall MAP scores of the opinion retrieval using these configurations are listed in Table 2.

Table 2. MAP scores from different system configurations

	Configuration	MAP
1	IR	0.1826
2	IR+OC	0.1740
3	IR+OC+NEAR	0.1828
4	IR+OC+NEAR+STCS	0.1938
5	IR+OC+NEAR+STCC	0.2052
6	IR+OC+STCS	0.1399
7	IR+OC+STCC	0.1337

In line 1 of table 2, the performance of IR component alone is strong (0.1826), comparing to the best multi-component configuration (0.2052 of line 5). The reason is that in a subjective information rich environment, such as the blog documents, the topic-relevant factual content itself is a good indication of the topic-relevant comments in the documents, having more query-relevant factual contents means higher chance of having query-relevant comments. In such a document collection, a traditional IR system can still have a reasonable performance in term of opinion retrieval effectiveness, even not considering subjective information at all.

In line 2, adding the opinion classification component decreases the MAP by about 4.9% with respect to line 1. This is because that just identifying the subjective contents, but without distinguishing the query-relevant opinions from query-irrelevant opinions, is not enough for opinion retrieval. The query-irrelevant opinions cause deterioration to the retrieval effectiveness.

The NEAR operator is designed to identify the query-relevant opinions. Adding the NEAR operator into the system brings the MAP back to 0.1828, slightly higher than that of line 1. This, together with line 2, indicates that query-relevant opinions are the actual subjective contents helping the opinion retrieval, but not any opinions.

The stcs and stcc opinion similarity functions calculate the similarity scores based on the amount of subjective information found in a document. Their performances heavily depend on the quality of the recognized subjective texts in the documents. In line 4 and 5 of table 2, stcs and stcc functions increase the MAP by 6.02% and 12.25% respectively, comparing to line 3. This shows that the amount of the query-relevant opinions in a document is a positive factor of the opinion similarity.

When comparing the score of line 4 to that of line 6, and the score of line 5 to that of line 7 respectively, we observe sharp decrease of MAP if the system does not capture query-relevant opinions. Removing the NEAR operator causes 27.81% and 34.83% of MAP decreases for stcs and stcc configurations respectively, which are much more than the decrease from line 3 to line 2. The reason is that in both lines 2 and 3, documents are ranked by using their similarities from the IR component. The NEAR operator filters out the documents that do not have query-relevant opinions. But it does not add any similarity to the remaining documents, and the ordering of the remaining documents in line 3 is the same as that in line 2. So the MAP decrease is relatively small. However, the cases of lines 6 and 4 are different. In line 6, after removing the NEAR operator, all the subjective sentences become query-relevant. But in line 4, only the subjective sentences qualified by the NEAR operator are considered as query-relevant. So in line 6, the scores of those query-irrelevant subjective sentences are also included into the opinion similarity scores. This causes significant changes in the similarity scores that are calculated by the *stcs* function, which results in big changes of the document ranking and therefore the big MAP decrease of line 6, compared to the MAP of line 4. Removing the NEAR operator from line 5 causes the same thing to happen in line 7. This again proves that the query-relevant opinions play a very important role in the opinion retrieval, and the NEAR operator is critical to the stcc/stcs similarity function.

The 0.2052 MAP from configuration 5 is higher than the highest MAP (0.1885) of the automatic, title-only runs that were reported in the TREC 2006 Blog Track [17]. This 0.2052 MAP equals to the highest MAP of all the automatic runs (including systems using title-description-narrative queries) reported in [17]. This indicates that our algorithm works well for opinion retrieval.

Table 2 shows that the IR, OC, NEAR and opinion similarity functions are all essential for our opinion retrieval system. Their cooperation maximizes the retrieval effectiveness.

6.2 Experiments Using More Statistics

In this experiment, we study the role of the opinionative feature sets in the opinion retrieval. We want to test whether larger opinionative feature sets, which result from using more training data, can be beneficial to the opinion retrieval. We use the configurations of 3, 4 and 5 from section 6.1 as the baselines. The

opinion classifiers of the baselines are trained by using the features selected from the training data of section 6.1. The comparing systems use the identical configurations of 3, 4 and 5, but the classifiers are trained with larger opinion feature sets, which result from more training data. Comparing to the training data collecting methods in section 6.1, the additional source epinions.com is used to collect more subjective sentences. Google is always used to collect both subjective and objective sentences. In addition, all 5 "subjective indicator phrases" are used. The SVM classifiers in the comparing systems get more training data than those in the baselines, since the new configuration uses the documents from more resources. Averagely each SVM classifier in the baseline system has 2,661 features. Each SVM classifier in the comparing system has 3,777 features on the average. We name the baselines and the comparing systems as follows:

1. **BL_ir**: Configuration (3) in section 6.1, baseline.
2. **BL_stcs**: Configuration (4) in section 6.1, baseline.
3. **BL_stcc**: Configuration (5) in section 6.1, baseline.
4. **NEW_ir:** BL_ir with SVM classifiers trained by larger feature sets, which are from larger training sets.
5. **NEW_stcs:** Identical to NEW_ir but use *stcs* function instead of *ir* function.
6. **NEW_stcc:** Identical to NEW_ir but use *stcc* function instead of *ir* function.

Table 3. Compare new configurations that use more features to baselines

	Baseline	MAP	New	MAP	Increase
1	BL_ir	0.1828	NEW_ir	0.2013	10.12%
2	BL_stcs	0.1938	NEW_stcs	0.2392	23.43%
3	BL_stcc	0.2052	NEW_stcc	0.2457	19.74%

Table 3 shows the MAP scores from the six configurations. All the "NEW" configurations that use larger feature sets get higher MAP scores than the corresponding baselines, regardless of what ranking strategies used. The increments are from 10% to 23%. The reason is that all the features used in the classifiers have biased distributions, which are either objective-oriented or subjective-oriented. As having more features means more characteristics of the objective language and subjective language are captured, better retrieval effectiveness is achieved.

6.3 Combined Similarity Functions

We have defined two combined similarity functions in formulas 5.4 and 5.5. In this experiment, we still use the "NEW" configurations of section 6.2, but using the formulas (5.4) and (5.5) for similarity calculation. We test them using values of 0.1, 0.3, 0.5, 0.7 and 0.9 respectively for both *a* and *b*. The MAP scores of these two configurations at various coefficient values are reported in Table 4. The "Avg. single MAP" shows the average MAP of the two single configurations used in a formula. For example, in formula 5.5, the average MAP of the *ir* configuration (0.2013 when a=1) and the *stcc* configuration (0.2457 when a=0) is 0.2235. The "Avg. Comb MAP" shows the average MAP of the five combined MAPs of a formula when the coefficient has values in (0, 1). For example, formula 5.5 have an average of 0.2596.

Table 4. MAP scores of the combined configurations

Formula 5.4		Formula 5.5	
Coefficient	Combination	Coefficient	Combination
a	NEW_ir_stcs	b	NEW_ir_stcc
0	0.2392	0	0.2457
0.1	0.2470	0.1	0.2582
0.3	0.2523	0.3	0.2605
0.5	0.2539	0.5	0.2643
0.7	0.2503	0.7	0.2593
0.9	0.2447	0.9	0.2559
1	0.2013	1	0.2013
Avg Single MAP	0.2203	Avg Single MAP	0.2235
Avg Comb. MAP	0.2496	Avg Comb. MAP	0.2596

In both formulas, all the combined MAPs are higher than the two single MAPs. The average MAP increases are about 13% for formula 5.4 and 16% for formula 5.5. The linear combination significantly improves the opinion retrieval effectiveness of our algorithm. The highest combined MAP for each formula appears when the coefficient is 0.5. This indicates that a simple half-to-half combination is good to maximize the power of the linear combination. The reason for this MAP increase in combined similarity functions is that the *ir* function uses the query-relevant factual contents to rank the retrieved documents, while the *stcc*/*stcs* functions use the query-relevant opinions to rank the documents. They capture different aspects of the documents. Put them together means more useful information is used in document ranking, which results in improved MAP.

6.4 Query Dependent Classifiers Vs. Query Independent Classifier

After we tested the query-dependent (Q-D) opinion classifiers in section 4.3.1, we want to test if the query-independent (Q-I) classifier has the same effect. The advantages of using a Q-I classifier are that (1) the system can have a simpler structure. The opinion classifier is constructed based on a set of training queries in multiple domains. So there is no need to create a Q-D classifier dynamically for each incoming query; and (2) the features in the Q-I classifier are determined independent of the queries to be processed, so that the documents can be indexed according to these features before query processing. Both (1) and (2) increase the query processing speed and reduce the retrieval response time, so that the efficiency of the system can be improved.

We build our first Q-I opinion classifier by using a use-all-training-data method. It utilizes the subjective and objective training documents of all 50 queries to construct one set of features, and build just one universal opinion classifier for the system. The systems that use this universal classifier are called the Q-I-50 systems. In practice, it is likely that an incoming query is not among the training queries. In order to estimate the performance of this situation, we build another type of Q-I classifier by using a leave-one-out method. It means, for a query *Q*, we merge the subjective (objective) documents of the other 49 queries to a single subjective (objective) set to build a SVM classifier, which is only used in the system when testing Q. This classifier is really independent of Q, because it does not use training documents of Q. We call this classifier the Q-I-49 classifier. Although we still end up having 50 classifiers, they are just for experimental purpose, to more accurately estimate the effectiveness of the query-independent opinion classification. A

realistic system will still have just one classifier. We call the systems using these classifiers the Q-I-49 systems.

All the previous "NEW" configurations in section 6.2 and 6.3 have their Q-D classifiers replaced by Q-I-49 and Q-I-50 classifiers. On the average, a Q-I-49/50 opinion classifier contains more than 70,000 features, which is about 17 times bigger than that of the Q-D classifiers (3,777 in section6.2). We repeat the experiments in Section 6.2 and 6.3 using these Q-I-49/50 configurations, and compare the results to those in section 6.2 and 6.3. The results are shown in table 5.

Table 5. Comparisons between configurations using Q-D and Q-I-49/50 opinion classifiers

		MAP		
		Q-D	Q-I-49	Q-I-50
1	NEW_ir	0.2013	0.2160	0.2101
2	NEW_stcs	0.2392	0.2422	0.2540
3	NEW_stcc	0.2457	0.2528	0.2600

Table 5 shows that, all the Q-I-49 and Q-I-50 systems achieve higher MAPs than those Q-D systems, The MAP increase for Q-I-49 system ranges from 1.25% (0.2392 to 0.2422) to 7.30% (0.2013 to 0.2160). Q-I-50 systems usually get higher MAP than Q-I-49 systems, except at line 1. The reason for the "Q-I scores higher than Q-D scores" is that more features tend to help in gaining retrieval effectiveness, in spite of the fact that many of the features are not relevant and not used for a specific query. We note that some of the 49 queries used to get the training data are in the same domain where the testing query belongs. For example, query 891 "intel" and 856 "macbook pro" are both in the "computer" domain. So even the training data is from "intel", the recognized features are still related to the computer domain. These features are still suitable for "macbook pro". So the domain, in which the testing query and some of the training queries both belong to, makes some of the features in an "independent training set" still relevant to the testing query. Clearly, the most relevant features for a query are obtained from the set of training documents of the same query. If that set is left out, some deterioration of retrieval effectiveness is expected. This is evident by comparing Q-I-49 and Q-I-50.

Table 6. Comparisons between the combined configurations using Q-D and Q-I-49/50 opinion classifiers

NEW_ir_stcs	Q-D	Q-I-49	Q-I-50
Avg.	0.2496	0.2522	0.2606
Max	0.2539 (a=0.5)	0.2566 (a=0.3)	0.2656 (a=0.3)
NEW_ir_stcc	Q-D	Q-I-49	Q-I-50
Avg.	0.2596	0.2649	0.2690
Max	0.2643 (b=0.5)	0.2687 (b=0.3)	0.2726 (b=0.5)

Table 6 compares the configurations that use combined similarity functions, when they use Q-D and Q-I opinion classifiers. Similar to the results of table 5, Q-I systems outperform the Q-D systems. But both increased and decreased MAPs are observed at the individual query level. Table 5 and 6 show that in our opinion retrieval system, using Q-I opinion classifiers is at least comparable to, or slightly better than using Q-D classifiers. The questions of how to choose right training queries and how many training queries to use are still worth further study.

6.5 Comparing to Related Works

In Table 7, we compare the best MAPs from our system to the best MAPs from various run categories in TREC 2006 Blog Track, and that from a state-of-art opinion retrieval system by Mishne [30] in 2007. All these systems are tested on the same TREC Blog data and query set. Our system and Mishne's system belong to the automatic, title-only run category (T). Other two automatic run categories are the title-description run (TD) and the title-description-narrative run (TDN). The title-only system uses only the query title as an original query. The title-description system uses terms in the query title and description as the original query. The title-description-narrative system uses terms in the query title, description and narrative as the original query.

Table 7. Comparisons between best MAPs from our algorithm and those from TREC 2006 Blog Track

TREC 2006		Mishne (T)	Q-D (T)	Q-I-49 (T)	Q-I-50 (T)
T	0.1885	0.2411	0.2643	0.2687	0.2726
TD	0.1887				
TDN	0.2052				

Table 7 shows that our algorithm, when using only the query titles as queries, has MAPs that are 28% to 32% higher than the highest TREC 2006 Blog Track MAP, even when those TREC systems utilize the description and narrative information. In practice, the title-only run is more realistic because users rarely provide the descriptions and narratives when they submit queries.

As introduced in the related works, Mishne [30] recently reported a state-of-art opinion retrieval system in 2007 that had a strong performance. This system achieved a MAP of 0.2411 when tested on TREC 2006 Blog data and the title-only queries. The best MAP of our systems is 13.1% higher than that of his system.

In this paper, we mainly focus on improving the retrieval effectiveness of our system, but have not addressed the efficiency issue in detail. Opinion retrieval is more complex than the traditional IR. Many opinion recognition processes are time consuming, such as the training and using of the SVM classifiers. It is expected that the opinion retrieval take more time. Improving the retrieval efficiency can be achieved by using the Q-I classifier instead of the Q-D, so that indexing the opinion features and the opinion classification of the document sentences can be done before query processing. Therefore, the IR step and the recognition of the ROS can be carried out in parallel for answering a query. Then the response time can be reduced.

In summary, our results show that: (1) Both query-relevant factual information and query-relevant opinions are necessary for an opinion retrieval system. (2) Generally more features in the opinion classifiers help the system improve retrieval effectiveness. (3) A linear combination of the *ir* similarity and the *stcc*/*stcs* similarity yields better results than the individual functions. (4) An universal query-independent opinion classifier is at least comparable to, or slightly better than the query-dependent classifiers in retrieval effectiveness. A Q-I classifier clearly can simplify the system and increase the retrieval speed.

7. CONCLUSIONS

In this paper, we presented a three-component opinion retrieval algorithm. The first component is an information retrieval module. The second one classifies documents into opinionative and non-opinionative documents, and keeps the former. The third

component ensures that the opinions are related to the query, and ranks the documents in certain order. We studied the effects of various parameters in our algorithm and empirically determined the best parameter configuration. The best MAP of our system is 28% to 32% higher than those of the best TREC 2006 Blog Track automatic runs, and is 13% higher than a state-of-art opinion retrieval system. We comment on how the entire system can be made efficient by utilizing query-independent features. In future studies, we plan to study in more detail on the choice of features and a better implementation of the NEAR operator.

8. ACKNOWLEDGMENTS

The authors thank the reviewers for their helpful comments. This work is supported in part by AOL under a research grant. The views of this paper are those of the authors, and do not represent those of AOL.

9. REFERENCES

[1] H Chernoff and E Lehmann. The use of maximum likelihood estimates in χ^2 tests for goodness-of-fit. *The Annals of Mathematical Statistics*. 1954.

[2] S Das, and M Chen. Yahoo! for Amazon: Extracting market sentiment from stock message boards. *APFA*, 2001.

[3] K Dave, S Lawrence, and D Pennock. Mining the Peanut Gallery: Opinion extraction and semantic classification of product reviews. *WWW'03*, 2003.

[4] A Esuli and F Sebastiani. Determining the semantic orientation of terms through gloss analysis. In *Proc. of CIKM*. 2005.

[5] V Hatzivassiloglou and K McKeown. Predicting the semantic orientation of adjectives. In *Proc. of ACL*. 1997.

[6] V Hatzivassiloglou and J Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. *COLING'00*. 2000.

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

[8] http://en.wikipedia.org

[9] M Hu and B Liu. Mining and Summarizing Customer Reviews. In *Proceedings of SIGKDD*. 2004.

[10] N Jindal and B Liu. Identifying Comparative Sentences in Text Documents. In *Proc. of the 29th SIGIR*. 2006.

[11] T Joachims. Making large-scale SVM learning practical. *Advances in Kernel Methods: Support Vector Learning*. 1999.

[12] Soo-Min Kim and Eduard Hovy. Determining the Sentiment of Opinions. In *Proc. of COLING*. 2004.

[13] N Kobayashi, R Iida, K Inui, and Y Matsumoto. Opinion mining on the Web by extracting subject-attribute-value relations. AAAI-CAAW'06, 2006.

[14] L-W Ku, L-Y Lee, T-H Wu, and H-H Chen. Major topic detection and its application to opinion summarization. In *Proc. of SIGIR*. 2005

[15] B Liu, M Hu and J Cheng. Opinion Observer: Analyzing and Comparing Opinions on the Web. In *Proceedings of the 14th WWW Conference*. 2005.

[16] S Liu, F Liu, C Yu, and W Meng. An Effective Approach to Document Retrieval via Utilizing WordNet and Recognizing Phrases. In *Proceedings of the 27th SIGIR*. 2004.

[17] I Ounis, M de Rijke, C Macdonald, G Mishne, and I Soboroff. Overview of the TREC-2006 Blog Track. In *Proceedings of TREC*. 2006.

[18] B Pang, L Lee and S Vaithyanathan. Thumbs up? Sentiment Classification Using Machine Learning Techniques. EMNLP'02, 2002.

[19] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*. 2005.

[20] A Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *Proc. of HLT/EMNLP*. 2005.

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

[22] R Tong. An operational system for detecting and tracking opinions in on-line discussions. *SIGIR 2001 Workshop on Operational Text Classification*, 2001.

[23] P Turney. Thumbs up or Thumbs down? Semantic orientation applied to unsupervised classification of reviews. *ACL'02*, 2002.

[24] Ellen M. Voorhees. Overview of the TREC 2005 Robust Retrieval Track. In *Proceedings of TREC-2005*.

[25] C Whitelaw, N Garg, and S Argamon. Using appraisal groups for sentiment analysis. In *Proc. of CIKM*. 2005

[26] J Wiebe, T Wilson, R Bruce, M Bell and M Martin. Learning subjective language. *Computational Linguistics*. 2004.

[27] Y Yang and J Pederson. A comparative study on feature selection in text categorization. In *Proc. of ICML*. 1997.

[28] H Yu and V Hatzivassiloglou. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. *EMNLP'03*, 2003.

[29] G. Mishne. Multiple ranking strategies for opinion retrieval in blogs. In TREC 2006, 2006.

[30] Gilad Mishne. Using Blog Properties to Improve Retrieval. In proceedings of ICWSM. 2007.

[31] Wei Zhang and Clement Yu. UIC at TREC 2006 Blog Track. In proceedings of 15th TREC. 2006.

[32] K Yang, N Yu, A Valerio and H Zhang. WIDIT in TREC 2006 Blog Track. In proceedings of 15th TREC. 2006.

[33] Douglas Oard, Tamer Elsayed, Jianqiang Wang, Yejun Wu, Pengyi Zhang, Eileen Abels, Jimmy Lin and Dagbert Soergel. TREC-2006 at Maryland: Blog, Enterprise, Legal and QA Tracks. In proceedings of 15th TREC. 2006.

[34] Wei Zhang, Shuang Liu, Clement Yu, Chaojing Sun, Fang Liu and Weiyi Meng. Recognition and Classification of Noun Phrases in Queries for Effective Retrieval. In proceedings of CIKM. 2007.

[35] Jinxi Xu and W Croft. Query Expansion Using Local and Global Document Analysis. In proceedings of SIGIR. 1996.