Evaluating Verbose Query Processing Techniques

Samuel Huston and W. Bruce Croft
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts, Amherst
Amherst, MA 01003
{sjh, croft}@cs.umass.edu

ABSTRACT

Verbose or long queries are a small but significant part of the query stream in web search, and are common in other applications such as collaborative question answering (CQA). Current search engines perform well with keyword queries but are not, in general, effective for verbose queries. In this paper, we examine query processing techniques which can be applied to verbose queries prior to submission to a search engine in order to improve the search engine's results. We focus on verbose queries that have sentence-like structure, but are not simple "wh-" questions, and assume the search engine is a "black box." We evaluated the output of two search engines using queries from a CQA service and our results show that, among a broad range of techniques, the most effective approach is to simply reduce the length of the query. This can be achieved effectively by removing "stop structure" instead of only stop words. We show that the process of learning and removing stop structure from a query can be effectively automated.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Query formulation

General Terms

Algorithms, Experimentation, Perfomance

Keywords

Verbose Queries, Query Reformulation, Black Box

1. INTRODUCTION

It has been observed that most queries submitted to search engines are short. For example, the average length of the queries in the MSN search log, a sample of about 15 million queries collected over one month, is 2.4 words [3]. A surprising number of queries, however, are longer. For example, in the MSN log, about 10% of the queries are five words or longer (not including extremely long run-on queries, such as automatically generated queries). The reason this is surprising is that current search engines do not perform particularly well with long queries. Long queries are important in other search applications, such as Collaborative Question Answering (CQA) services and certain vertical search domains, such as patent searches.

Long queries can potentially express more complex information needs and they can provide much more context for ranking algorithms than the one-word queries used as examples in many papers (e.g., "java" and "jaguar"). Given their potential, there has been some recent research on techniques for improving search effectiveness with long queries. Bendersky and Croft [2], for example, show that key concepts can be extracted from long queries and used to improve ranking. Lease et al [14] described a technique for improving ranking by weighting terms in long queries. Kumaran and Carvalho [12] use learning techniques to rank subsets of the original long query based on quality prediction measures. These and some other similar studies have used TREC collections and open-source search engines to study the effectiveness of query processing techniques for long queries.

In this paper, we focus on identifying query processing techniques that can be used to improve the effectiveness of long queries where the search engine is a "black box". As such, we have limited ourselves to techniques that generate queries that a standard search engine application programming interface (API; in our case, the Yahoo! API and the Bing API) is capable of processing. This means that some techniques, such as term weighting, cannot be used because they are not supported by the API. There is some risk in this decision as it is possible that the search engine will attempt to learn from the submitted queries or that our results will reflect the query processing done by the search engine. To reduce these risks, we used two different search engines for our experiments. We also attempted to limit the learning capability of each search engine by ensuring that the links returned were never 'clicked.' In order to reduce the chance that the search engine is modified during our search experiments, we ensure that all queries produced by applying these pre-processing techniques are submitted to the search APIs within a very short time span. We note that "black box" experiments are quite common in the literature and are becoming more important in the search industry. One of our goals is to make the assumptions and limitations underlying these experiments more explicit.

In addition to limiting our query processing techniques

SIGIR'10, July 19–23, 2010, Geneva, Switzerland.

to account for the black box search environment, we also restrict our study to long queries that form a clause or sentence containing a verb. Bendersky and Croft [3] described a classification of long queries and found that queries containing verbs made up about 15% of the long queries in the MSN query log. Our interest in these type of queries stems from the fact that their structure allows for more linguistic processing and that they are representative of the queries that people use in less constrained search environments, such as a CQA service. *Verbose* queries are long queries in which people use many words to say what could have been expressed in a few keywords. An example of a verbose query is "Would the meteor shower that hits this weekend be better to watch tonight or tomorrow night?", taken from Yahoo! Answers[1]. A number of the words in this query do not help to retrieve relevant documents, but the query as a whole conveys considerably more about the information needed or the user's intent than a query such as "meteor shower". Verbose queries may also include simple "wh-" (who, what, when, where) or "factoid" queries. However, because this subset of verbose queries has been studied extensively in the TREC QA (Question Answering) track and have had specific techniques developed for them, we have omitted them from this work.

The query transformation techniques we consider include stopword removal, phrase detection, key concept identification, and stop structure removal. Each of these techniques is applied individually and in combination to the verbose queries in our test sets. We measure the performance of the original queries and of the queries produced by these techniques and compare the results to evaluate the techniques' effectiveness. The most effective single technique found in this first round of experiments was stop structure removal.

A stop phrase has been defined as a phrase that does not provide any information on the user's information need [6]. We define a *stop structure* as a stop phrase that begins at the first word in a query. Though it may be possible to effectively remove many stop structures from search engine queries using static lists, similar to those used for stopwords, doing so may inadvertently remove relevant words from the query. To address this problem we describe a sequential classifier that enables us to automatically identify the stop structure in each query. We show that the queries produced by this classifier perform similarly to those produced by manually identified stop structure removal.

In the next section we discuss related work. We describe how the test sets of queries were created in section 3. In section 4, we describe the query processing techniques used. We describe the experimental setup and the results from the retrieval experiments in section 5. The automatic stop structure classifier is described and analyzed in section 6. In section 7 we conclude and discuss possible future work.

2. RELATED WORK

There has been a large amount of recent work on query transformation techniques. Much of this work focuses on the accurate, efficient extraction of linguistic features from queries. Bergsma and Wang [4] present a method of learning noun phrase segmentations. Tan and Peng [16] present a generative model based method of query segmentation. Guo et al [8] present a Weakly Supervised Latent Dirichlet Allocation (WS-LDA) based method of identifying named entities in queries. Each of these studies evaluate their performance over samples of commercial query logs. It should be noted that these studies do not show how the extracted noun phrases or query segments might be used in a retrieval system, either internally or externally as a pre-processing stage.

There is some recent work on query processing that evaluates results using a "black box" approach. For example, Gu et al [9] describe a CRF-based framework for applying several types of transformations to a query. These transformations are tested on queries sampled from a commercial search engine log. Both the original and transformed queries are submitted to a commercial search engine, and the top retrieved documents are assessed for relevance. Although there are some similarities to our study, we focus on verbose queries and processing techniques that are appropriate for longer queries, including automatic stop structure removal.

3. TEST QUERIES

In order to carry out experiments with query processing techniques, we needed a test collection of verbose queries. Rather than use the TREC description queries that have been studied in other recent papers (e.g., [2],[14]), we decided to use queries from the Yahoo! Answers CQA service. We believed these queries would be less artificial than TREC queries and less constrained than queries from a web search query log, where users tend to think in terms of keywords because that is what works well with current search engines. We would expect, therefore, that many of the CQA queries will not work well without query processing.

In order to construct a sample of verbose queries, we extracted questions from a crawl of Yahoo! Answers and filtered the results to identify queries which match the desired type. As mentioned above, the desired queries consist of clauses or sentences containing verbs, but are not standard "factoid" or QA questions. In Bendersky and Croft's study [3], these were called *non-composite-verb* queries, where the "non-composite" indicates that the query is not just a simple combination of shorter web-style queries.

Our purpose in extracting queries is to construct two non-overlapping sample sets of 50 queries, such that one set is composed of more "difficult" queries, and the other set is composed of a representative sample of this type of query. In order to construct the "difficult" query set, candidate queries were submitted to a web search engine (Yahoo! Search). If the search yielded a relevant document in the top position, then the query was rejected, otherwise we added the query to our test set.

In this paper we refer to the set of "difficult" queries as *Set 1*. The representative set of verbose queries is referred to as *Set 2*.

Set 1 is used to study the query processing techniques that could have the most impact when current search engines fail to perform well. *Set 2* is used to test that the same query processing techniques are beneficial over a more representative sample query set.

By inspecting the query log we found that capitalization and punctuation appeared to be optional for this type of query. Accordingly, we removed all punctuation from the queries in order to ensure that the performance of our techniques would not be dependent on correct grammar or capitalization. We view spelling correction as a necessary step

[1]http://answers.yahoo.com/

for any verbose query processing system. Therefore we modified the queries so that all words were correctly spelled.

4. TECHNIQUES

4.1 Stopword Removal

Removing stopwords has long been a standard query processing step [7]. We used three different stopwords lists in this study: the standard INQUERY stopword list [1], as well as two query-stopword lists derived from the Yahoo! Answers query log. We chose to construct new stopword lists because the language that is used in some of the test queries is more informal than the sample of English text from which the INQUERY stopword list is derived.

Lo, He and Ounis [17] present several methods of automatically constructing a collection dependent stopword list. Their methods generally involve ranking collection terms by some weight, then choosing some rank threshold above which terms are considered stopwords. Their weighting methods all produce similar retrieval performances, but an inverse document frequency (IDF) based weighting scheme is the most effective.

In accordance with their findings, we applied an IDF based weighting scheme to the set of verbose queries in the Yahoo! Answers query log. As a result we were able to construct two stopword lists which might be more appropriate for these queries. The two stopword lists were extracted by taking the top 100 and 200 ranked words from this ordering. It should be noted that words are rarely repeated in a single query. Thus, in this particular case, sorting by term frequency is almost rank equivalent to sorting by inverse document frequency.

This process identified words such as "help", "find", and "know" which do not occur in the INQUERY stopword list, but are not necessarily useful search words.

The key problem in trying to construct a query-stopword list is that common query phrases such as "high blood pressure" occur as frequently as some stop phrases. These frequent phrases mean that words such as "blood" and "pressure" occur in the query log as frequently as the words "did" or "am".

We chose to apply these stopword lists by removing all words in the query which occur on the stopword list. For example, if using the INQUERY stopword list, and given the query:

"Can i work while study in Europe"

this technique produces the query:

"work study Europe"

It is worth noting that this technique can significantly change the meaning of the query. For example when applying the INQUERY stopword list to the query:

"i would like to know the origin
of the phrase to be or not to be"

the resulting query is:

"know origin phrase"

4.2 Noun Phrase Detection

By definition, a verbose query is composed by the user as a sentence or phrase. This allows us to apply a variety of linguistic tagging and chunking techniques to the query. Additionally we know from previous work that extracting noun phrases from the query can help identify the key concepts within the query [2, 18, 10]. We used the MontyLingua toolkit [15] to automatically identify noun phrases.

Given that we are using a search engine as a black box, we are limited in how we can use the identified noun phrases contained within the query. We are unable to assign weights to terms or phrases according to confidence or importance. Using the black box interface, we are only able to identify important phrases using quotation marks.

There are several methods by which the identified noun phrases can be communicated to the search engine within this limited interface. We show results for two such methods. The first method wraps each of the detected noun phrases in the query in quotation marks; no words are removed from the query. The second method removes all words which are not part of some detected noun phrase and quotation marks are not used.

For example, 3 noun phrases: *'me'*, *'a name'* and *'my next horror script'*, are detected in the query:

give me a name for my next horror script

Using the first method we would produce the query:

give "me" "a name" for "my next horror script"

Using the second method we would produce the query:

me a name my next horror script

4.3 Key Concept Detection

The term key concept in this context is a short set of sequential words that express an important idea contained within the query. As presented by Bendersky and Croft [2], the identification of key concepts in long queries can produce performance improvements in retrieval engines. Their research demonstrated that key concepts in TREC queries can be automatically identified with around 80% accuracy.

For this study, we use the same type of classifier as Bendersky and Croft. The classifier is based upon the AdaBoost.M1 meta-classifier using C4.5 Decision Trees. The features used in this classifier included: GOV2 collection term frequency, inverse document frequency, residual inverse document frequency, weighted information gain, google n-grams term frequency, and query frequency. These features are detailed in their paper, [2]. The GOV2 query set was used to train this classifier.

Key concepts are structurally similar to noun phrases and thus present similar problems for utilizing them. In order to effectively communicate key concepts to the search engine, therefore, we again employ the same two methods. The first method wraps the detected key concepts in quotation marks. The second method removes all words that are not part of a given key concept.

Given the query:

i read that ions cant have net dipole moments why not

The key concepts "i", "ions" and "net dipole moments" were identified by the classifier. The first method outlined above would produce the query:

"i" read that "ions" cant have "net dipole moments" why not

and the second method would produce:

i ions net dipole moments

4.4 Stop Structure

Removal of *stop phrases* was originally presented by Callan and Croft as a query processing technique [6]. They define a stop phrase as a phrase which provides no information about the information need. One example they give is "find a document". Within their study they created a static list of stop phrases by inspecting 50 TIPSTER queries. This list was reused in later studies using TIPSTER [5].

The majority of our test queries contained one relatively large stop phrase at the start of the query. We call these stop phrases *stop structure*. That is, a *stop structure* is a stop phrase which begins at the first word in the query. For this study other stop phrases in the query are considered semantically meaningful; we did not remove them.

Importantly, stop structure often contains words which are not stopwords, but which nonetheless do not add any meaning to the query. For example "My husband would like to know more about cancer" This information need could be equivalently served by the query "cancer". As in the case of stopwords, stop structure removal carries some inherent risk. In the above query, one could say that there is implied information contained within the term "husband". That is, the underlying information need may correspond to specific types of cancers that are common among males.

However, within the framework of this study we have access to additional information about the user's information need. We used the user-selected answer corresponding to their query to guide the manual identification of stop structure. In this way, we eliminated the risk of losing meaningful terms through the removal of stop structure. Later, we will show that automatic, non-answer guided stop structure removal can closely approximate the retrieval performance of answer-guided manual stop structure removal.

Similar to stopwords, stop structure is removed from queries prior to submission to a search engine. For example given the query:

if i am having a lipid test can i drink black coffee

The stop structure "if i am having a" is removed, leaving:

lipid test can i drink black coffee

Note that this query has another stop phrase, "can i", which is not removed.

5. RETRIEVAL EXPERIMENTS

5.1 Setup

The aim of these experiments is to see which of the above techniques, if any, has the greatest effect on retrieval performance. As mentioned above we choose to use two commercial search engines as black box search engines, as this matches the limitations of a meta-search engine designer. For this study we chose to use Yahoo! Search, and Bing as 'black boxes'. Yahoo! BOSS API and Bing API 2.0 were used to submit queries and retrieve documents. 10 results were collected for each query from each search engine.

There are two problems which may arise through the use of commercial search engines for a scientific study. First the search engine may learn how to respond to our particular set of test queries. Second the search engine may be modified at any time. In order to limit the amount of learning that each search engine would be able to do over our set of test queries, we ensures that no returned links were clicked for any of the queries.

Since the search engines we are using are commercial, we won't necessarily know if the underlying system changes at any time. In order to minimize this risk we chose to submit all queries generated by all of the above techniques to both APIs within a single short search session. This search session occurred on September 29th, 2009.

After all queries were submitted, all returned pages were manually judged for relevance. These judgements were guided by the answers provided for the queries in the CQA system. A three-valued judgement system was used. The three values were 'not relevant', 'partially relevant', and 'relevant', denoted by the numerical values {0,1,2} respectively.

It should be noted that both search engines occasionally returned the page in Yahoo! Answers from which the original query was gathered. These pages were removed from the search results, and the remaining pages re-ranked. This adjustment was made in order to avoid skewed results in favor of the original queries.

We show the evaluation metrics normalized discounted cumulative gain (nDCG) at ranks 5 and 10 for each processing technique. The normalization constant was computed from the sum of all annotations for each query. Thus the values are comparable across query processing techniques, the two search engines, and the subsequent retrieval experiments described below.

5.2 Retrieval Experiments Results

Table 1 shows the retrieval results for all of the query processing techniques when applied to query set 1 using the Yahoo! and Bing search engines. The results from the two search engines are very similar in terms of relative effectiveness, confirming that inconsistencies are unlikely to have been caused by internal search engine processing. The use of quotations in formulating queries is clearly not effective. Both noun phrase and key concept identification produce significant improvements, when combined with stopword removal. The most effective technique, however, is the removal of stop structure. Manual removal of these words resulted in consistent and very significant improvements for both NDCG measures and in both search engines.

Manual removal of stop structure was guided by the user's selected answer, so we view this as an oracle result under the best possible circumstances.

Interestingly, the removal of any remaining stopwords from the queries after removing stop structure, degrades the retrieval performance of the queries Within almost all of the queries produced through the removal of manual stop structure semantically meaningless terms are present. The performance degradation stems from the removal of semantically meaningful terms. For example given the query:

for a year ive been getting some tightening from within my chest

Removal of manual stop structure produces the query:

tightening from within my chest

Clearly "from" and "my" could be removed without changing the meaning of the query. However, removing the INQUERY

	Query Test Set 1 Yahoo! API		Query Test Set 1 Bing API	
	nDCG@5	nDCG@10	nDCG@5	nDCG@10
Original	0.1760	0.1573	0.1939	0.1875
INQUERY Stopword List	0.2263^{+}	0.2060^{+}	0.2530^{+}	0.2152
Query Stopword List 1	0.2116	0.1900	0.2551^{+}	0.2266
Query Stopword List 2	0.1844	0.1846	0.2142	0.1952
Quoted Noun Phrases	0.0854^{-}	0.0735^{-}	0.0947^{-}	0.0763^{-}
Quoted Noun Phrases + Stopwords	0.1172^{-}	0.1071^{-}	0.1384	0.1262^{-}
Quoted Key Concepts	0.1217^{-}	0.1110^{-}	0.1261^{-}	0.1201^{-}
Quoted Key Concepts + Stopwords	0.1401	0.1340	0.1749	0.1536
Only Noun Phrases	0.2164	0.1975	0.2305	0.2107
Only Noun Phrases + Stopwords	0.2808^{+}	0.2530^{+}	0.2854^{+}	0.2507
Only Key Concepts	0.2344	0.2165^{+}	0.2612	0.2447
Only Key Concepts + Stopwords	0.2851^{+}	0.2531^{+}	0.2894^{+}	0.2684^{+}
Manual Stop Structure	0.3604^{+}	0.3298^{+}	0.3588^{+}	0.3274^{+}
Manual Stop Structure + Stopwords	0.3280^{+}	0.3042^{+}	0.3798^{+}	0.3371^{+}

Table 1: Performance of preprocessing techniques from the first search session for query test set 1. nDCG@5 and nDCG@10 are shown for both search APIs. Paired t-tests were performed between each result shown and the baseline (Original), results which show significant improvements, (p-value < 0.05) are marked $^{+}$. Similarly results which significantly degraded performance, with a p-value less than 0.05, are marked $^{-}$.

	Query Test Set 2 Yahoo! API		Query Test Set 2 Bing API	
	nDCG@5	nDCG@10	nDCG@5	nDCG@10
Original	0.3355	0.2926	0.2757	0.2400
INQUERY Stopword List	0.3610	0.3311	0.3356^{+}	0.3157^{+}
Query Stopword List 1	0.3289	0.3009	0.3359	0.3172^{+}
Query Stopword List 2	0.3118	0.3020	0.3546^{+}	0.3329^{+}
Manual Stop Structure	0.4288^{+}	0.3805^{+}	0.4201^{+}	0.3815^{+}
Manual Stop Structure + Stopwords	0.4071^{+}	0.3789^{+}	0.4054^{+}	0.3882^{+}

Table 2: Performance of preprocessing techniques from the first search session for query test set 2. nDCG@5 and nDCG@10 are shown. Paired t-tests were performed between each result shown and the baseline (Original), results which show significant improvements, (p-value < 0.05) are marked $^{+}$.

stopwords produces the query:

tightening chest

Both search engines returned websites which advertised exercise regimes for this query. Similarly the performance of the stopword based techniques was hindered by this type of inadvertent removal of semantically meaningful terms.

Recall that query test set 1 was sampled such that the queries are more difficult for commercial search engines. We repeated the experiments with stopword and stop structure removal using the second set of queries. These results are shown in Table 2. This second set of experiments are designed to test the performance of the removal of stop structure over a more representative sample of verbose queries.

These results show that stop structure removal is indeed very effective. Given these, very significant effectiveness improvements from manual stop structure removal, in the remainder of this paper, we focus on describing and evaluating an automatic method for classifying and removing query stop structure.

6. AUTOMATIC STOP STRUCTURE CLASSIFICATION

We have defined *stop structure* to be a stop phrase beginning at the first word in the query. It it important to note that a stop structure can have many variations without significantly changing its meaning. Indeed, any phrase that does not provide some insight into the underlying information need may be considered stop structure for some particular query. Additionally, it is possible that the stop structure from one query may not necessarily be stop structure in another query. For example "please tell me why" may be identified as stop structure in the query:

please tell me why the sky is blue

But it would not be considered stop structure in the query:

please tell me why song lyrics

So we can see that using a single static stop structure list is not an appropriate approach to stop structure removal.

Therefore, we have developed a method of automatically classifying stop structure within some input query. The purpose of our classifier is to identify each of the words in each of the queries as a "stop structure term" or a "query term".

It is natural to formulate this type of problem as a sequential binary class tagging problem.

6.1 Features

In order to obtain the best possible classifier a range of features were extracted for term in each query. Broadly these features fall into two categories; multinomial and numerical. The features extracted for each term in a query were then used as input to the classifiers which are discussed in the next section.

First I will outline the multinomial features generated for each term in the input query. Using the MontyLingua toolkit [15], we extract the part of speech tag for each term (NN, VB, etc) The next feature generated was the position of the term (1,2,3,4,5,...) in the query. Two binary features were also extracted: whether or not the term is present in the INQUERY stopword list, and whether or not there is any non-stopwords in the current query which precede the current word.

The numerical features were extracted from three TREC collections, and from three query logs. We used the WT10G, GOV, and GOV2 TREC collections. The query logs we used are: the Yahoo! Answers CQA log, Wondir CQA log, and the MSN query log. We found that each of these collections and query logs conveyed some unique information to the classifier. We believe that this is due to the vocabulary mismatch between the test queries and any one of the collections or query logs.

All three TREC collections are composed of web documents, the GOV and GOV2 collections were limited to the '.gov' domain, while the WT10G was not limited to any particular domain. Further details can be found on the TREC Web Test collections website. [2]

The Yahoo! Answers CQA log consists of 216,563 user submitted question and answer sets. The log was constructed by a web crawl of the Yahoo! Answers website [3]. The Wondir CQA log consists of 401,560 user submitted question and answer sets from the Wondir community question answer service. The MSN query log consists of 15 million queries submitted to the MSN search engine during a period of one month.

We define subsets of each of the query logs were created based on the length of the queries; the first subset contains very short queries ($\leq$ 2 words), the second contains short queries ($\leq$ 4 words) and the final subset contains long queries ($>$ 4 words). Our hope is that features extracted from these subsets will enable a classifier to distinguish between commonly searched phrases and common stop structures. A common search phrase should have a higher frequency amongst the short or very short queries, while a common stop structure should have a lower frequency in these subsets. Within the long subsets, common query phrases and stop structures may have similar, relatively high frequencies.

Three types of numeric features were extracted for each term in a query; term frequency, bi-term frequency, and inverse document frequency (IDF). Term frequency refers to the number of times the term occurs in the collection; $f(t_i)$. Bi-term frequency referse to the number of times the term and it's predecessor occurs in the corpus; $f(t_i|t_{i-1})$. IDF refers to the inverse document frequency; $idf(t_i) = \log \frac{|D|}{d_f}$. Where the document frequency, d_f, is the number of documents which contains the term, and $|D|$ is the number of documents in the collection.

We extracted all three numeric features from each of the TREC collections. Term frequencies and bi-term frequencies were extracted from each of the three query logs and each of the nine query log subsets.

We also experimented with features based on phrase chunk tags and n-word frequencies. However these features were found to be detrimental to the classifier's performance. This may have stemmed from the inaccuracy of the phrase chunks, and corpus sparsity problems for the frequency of longer phrases.

6.2 Classification Evaluation

As has been mentioned above, the purpose of this classifier is to mark each of the words in the queries as a "stop structure term" or a "query term". We experimented with two implementations of sequential taggers; CRF++ and YamCha. CRF++ is an open source implementation of Conditional Random Fields for sequential tagging, based on work done be Lafferty, McCallum and Pereira [13]. YamCha (Yet Another Multipurpose CHunk Annotator) is a sequential tagger based on support vector machines [11].

The training set used for each of these classifiers was composed of 100 new verbose queries identified from the Yahoo! Answers query log. These newly identified queries have similar properties to the two sets of queries already identified. The stop structure in these queries was manually identified in a similar manner to the test sets of queries. The test sets for this classifier are the two sets of queries which we have already identified for this study.

The CRF++ classifier is constructed such that all features must be multinomial. This means that numerical features are unsuitable for this classifier. In order to convert the frequency-based features detailed above into multinomial features, we applied binning after taking the log of each numerical value. After taking the log of the frequency values the numerical features ranged from 0 and 20. The most effective binning method we were able to find segmented the number plane at multiples of 5, with an extra bin for values $0 < x < 1$. This method produced 6 bins; (0,≤ 1,≤ 5,≤ 10,≤ 15,≥ 15).

The best performance from the YamCha classifier that we were able to obtain used a context window of 3 terms; the previous term, the current term and the next term. The previously assigned tag was also used as a feature. Using larger window sizes degraded performance. Similarly the best performance for the CRF++ that we were able to obtain used a context window of 3 terms.

We used two types of evaluation metrics to measure the performance of each classifier. First we used precision, recall, and accuracy of the tags applied to each term in the queries. A true positive, with respect to precision and recall, refers to a semantically meaningful term being correctly classified as a 'query term'.

Recall that stop structure begins with the first word in the query, and extends to the first semantically meaningful term, or 'query term'. The second measure is the mean squared error distance (MSE) of the position of the first 'query term'. This metric is intended to measure the error distance of the boundary between stop structure terms and query terms.

[2] http://ir.dcs.gla.ac.uk/test_collections/

[3] http://answers.yahoo.com/

	Test Set 1				Test Set 2			
	Precision	Recall	Accuracy	MSE	Precision	Recall	Accuracy	MSE
CRF++	0.891	0.825	0.816	5.1	0.930	0.867	0.866	2.68
Yamcha	0.915	0.921	0.894	2.64	0.938	0.926	0.905	1.86

Table 3: Classification results for two automatic stop structure classifiers. Precision, recall, and accuracy refer to per word classification accuracy. Precision and recall metrics refer to a correctly classified semantically meaningful query words. Mean squared error distance (MSE) refers to the distance between the first semantically meaningful query word, and the first classified semantically meaningful word.

	Query Test Set 1 Yahoo API		Query Test Set 1 Bing API	
	nDCG@5	nDCG@10	nDCG@5	nDCG@10
Original	0.1963	0.1718	0.2043	0.1803
Stopword List	0.2264	0.2011	0.2354	0.2162
Manual Stop Structure	0.3705^{+}	0.3338^{+}	0.3330^{+}	0.3059^{+}
Manual Stop Structure + Stopwords	0.3631^{+}	0.3177^{+}	0.3551^{+}	0.3267^{+}
Classified Stop Structure	0.3810^{+}	0.3412^{+}	0.3176^{+}	0.2893^{+}
Classified Stop Structure + Stopwords	0.3530^{+}	0.3177^{+}	0.3299^{+}	0.3038^{+}

Table 4: Performance of preprocessing techniques from the automatically classified stop structure focused retrieval experiments over query test set 1. Performance metric nDCG at 5 and at 10 are shown. Paired t-tests were performed between each result shown and the baseline (Original), results which show significant improvements, (p-value < 0.05) are marked $^{+}$.

	Query Test Set 2 Yahoo API		Query Test Set 2 Bing API	
	nDCG@5	nDCG@10	nDCG@5	nDCG@10
Original	0.3689	0.3394	0.3249	0.2710
Stopword List	0.3914	0.3770	0.4444^{+}	0.3821^{+}
Manual Stop Structure	0.4661^{+}	0.4312^{+}	0.4916^{+}	0.4507^{+}
Manual Stop Structure + Stopwords	0.4873^{+}	0.4525^{+}	0.5291^{+}	0.4782^{+}
Classified Stop Structure	0.4083	0.3923	0.4628^{+}	0.4253^{+}
Classified Stop Structure + Stopwords	0.4442^{+}	0.4148^{+}	0.5047^{+}	0.4692^{+}

Table 5: Performance of preprocessing techniques from the automatically classified stop structure focused retrieval experiments over query test set 2. Performance metric nDCG at 5 and at 10 are shown. Paired t-tests were performed between each result shown and the baseline (Original), results which show significant improvements, (p-value < 0.05) are marked $^{+}$.

Results for classification evaluation are shown in Table 3. We are most interested in maximizing recall within this classifier. A high recall value means that all semantically meaningful terms are retained within the processed query. That is not to say that precision is not valuable; higher precision should dramatically improve the performance of the query, as semantically meaningless words are removed from the processed query.

Due to the effectiveness of the YamCha classifier, the output of this classifier was used as the automatically classified stop structure in the following retrieval experiments.

6.3 Retrieval Experiments

We performed these retrieval experiments in a similar way to the previous retrieval experiments in section 5. The aim of these experiments is to evaluate the performance of the queries produced by removing automatically identified stop structure.

Because the commercial search engines we are using may have changed in the months between this search session and the previous search session, we chose to re-submit queries produced by previously analyzed pre-processing techniques. In particular, the original query, and techniques involving the removal of INQUERY stopwords, and the removal of manually identified stop structure were re-submitted as part of these experiments. Queries produced by these techniques and queries produced through the automatic removal of stop structure were submitted to Yahoo! API and Bing API during a search session on January 9th, 2010.

The retrieval performance for each of the processed queries as submitted in the second search session is shown in Tables 4 and 5. We show nDCG@5 and nDCG@10 for each processing technique. We use the same normalizing factor used in the above retrieval experiments, thus these results are directly comparable to the results in the previous retrieval experiments.

Comparing these results to the results of the previous session, we can see that the performance of these queries and pre-processing techniques have improved somewhat. in the months since the last search session. However, we can see that the relative performances of the processing techniques has not significantly changed. Queries produced by remov-

ing stop structure still significantly outperform the original queries and the queries with stopwords removed.

We can infer from these results that stop structure can be effectively automatically classified. When compared with the manually identified stop structure, similar performance increases over the baseline are achieved. The differences between the manually identified stop structure and the classified stop structure generally stemmed from misclassified semantically meaningful query words. That, is some meaningful terms are incorrectly classified as part of the stop structure. For example, in the query *"define turf toe as in football"*, the automatic stop structure classifier identified *"define turf"* as stop structure, thereby missing documents on the medical condition *"turf toe"*.

7. CONCLUSION

We have shown that pre-processing techniques are a viable option to improve the performance of verbose queries for commercial web search engines. In particular, we found that the removal of stop structure, a stop phrase beginning at the first word in the query, can dramatically improve the performance of the query.

We also investigated methods of automatically identifying stop structure within a given query. Using a YamCha based classifier, we were able to classify stop structure with high degree of accuracy. It was also shown that the retrieval performance of automatically identified stop structure closely matches the performance of manually identified stop structure.

Future work may include investigating the automatic removal of stop phrases that may occur at any position within a verbose query. This is based on the observation that it is possible to formulate queries which contain stop phrases which do not occur at the beginning of the query.

It would also be interesting to investigate methods of combining the results of several partial query searches. It is possible that some rank fusion methods might produce better results by applying several pre-processing techniques to the query then combining the results from several searches.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-0534383. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

8. REFERENCES

[1] J. Allan, W. Croft, D. Fisher, M. Connell, F. Feng, and X. Li. Inquery and TREC-9. In *Proc. of TREC-9*, pages 551–562, 2000.

[2] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *Proc. of SIGIR '08*, pages 491–498, New York, NY, USA, 2008. ACM.

[3] M. Bendersky and W. B. Croft. Analysis of long queries in a large scale search log. In *Workshop on Web Search Click Data (WSCD 2009)*, 2009.

[4] S. Bergsma and Q. Wang. Learning Noun Phrase Query Segmentation. In *Proc. of EMNLP-CoNLL 2007*, pages 819–826, 2007.

[5] J. Callan, W. Croft, and J. Broglio. TREC and TIPSTER experiments with INQUERY. *Information Processing and Management*, 31(3):327–343, 1995.

[6] J. P. Callan and W. B. Croft. An evaluation of query processing strategies using the TIPSTER collection. In *Proc. of SIGIR '93*, pages 347–355, New York, NY, USA, 1993. ACM.

[7] B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company, USA, 2009.

[8] J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *Proc. of SIGIR '08*, pages 267–274, New York, NY, USA, 2009. ACM.

[9] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *Proc. of SIGIR '08*, pages 379–386, New York, NY, USA, 2008. ACM.

[10] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proc. of EMNLP 2003*, volume 2, pages 16–223, 2003.

[11] T. Kudoh and Y. Matsumoto. Use of support vector learning for chunk identification. In *Proc. of CoNLL-2000*, page 144. Association for Computational Linguistics, 2000.

[12] G. Kumaran and V. R. Carvalho. Reducing long queries using query quality predictors. In *Proc. of SIGIR '09*, pages 564–571. ACM, 2009.

[13] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML 2001*, pages 282–289, 2001.

[14] M. Lease, J. Allan, and W. B. Croft. Regression Rank: Learning to Meet the Opportunity of Descriptive Queries. In *Proc. of ECIR 2009*, pages 90–101, 2009.

[15] H. Liu. MontyLingua: An end-to-end natural language processor with common sense, 2004.

[16] B. Tan and F. Peng. Unsupervised query segmentation using generative language models and Wikipedia. In *Proc. of WWW '08*, pages 347–356, New York, NY, USA, 2008. ACM.

[17] R. Tsz-Wai Lo, B. He, and I. Ounis. Automatically building a stopword list for an information retrieval system. In *Journal on Digital Information Management: Special Issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR)*, volume 5, pages 17–24, 2005.

[18] J. Xu and W. Croft. Query expansion using local and global document analysis. In *Proc. of SIGIR 1996*, page 11. ACM, 1996.