Learning Effective Ranking Functions for Newsgroup Search

Wensi Xi[1]
[1]Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA, 24060, U.S.A.
xwensi@vt.edu

Jesper Lind, Eric Brill[2]
[2]Microsoft Research, One Microsoft Way, Redmond, WA, U.S.A.
{jesperl, brill}@microsoft.com

ABSTRACT
Web communities are web virtual broadcasting spaces where people can freely discuss anything. While such communities function as discussion boards, they have even greater value as large repositories of archived information. In order to unlock the value of this resource, we need an effective means for searching archived discussion threads. Unfortunately the techniques that have proven successful for searching document collections and the Web are not ideally suited to the task of searching archived community discussions. In this paper, we explore the problem of creating an effective ranking function to predict the most relevant messages to queries in community search. We extract a set of predictive features from the thread trees of newsgroup messages as well as features of message authors and lexical distribution within a message thread. Our final results indicate that when using linear regression with this feature set, our search system achieved a 28.5% performance improvement compared to our baseline system.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information search and retrieval;

General Terms
Algorithms, Experimentation

Keywords
Newsgroup search, information retrieval, machine learning, linear regression, support vector machines.

1. INTRODUCTION
Web communities are web virtual spaces where people can freely discuss anything. Usenet is a web broadcasting service built for community discussions. People post to newsgroups to ask questions, answer questions, or partake in discussions. Search is a highly desirable feature on top of newsgroup archives, to enable users to search the message collections for information they need. Studies have shown that the vast majority of people who interact with an online community do it passively by browsing/searching archived discussions rather than directly participating in the discussions themselves.

Unfortunately, the techniques that have proven successful for searching document collections and the Web are not ideally suited to the task of community archive search. Compared to Web pages, newsgroup articles are typically much shorter, and do not have rich mark-up that can help determine term relevance. They also have a very different topological relation to other messages in the collection, making cues such as in-link analysis and anchor text impossible to utilize in this context.

In this paper, we explore whether a set of features unique to newsgroup discussion threads can be used to learn an effective ranking function that can be deployed in a newsgroup search engine to help people find relevant information.

The features we examined are based on attributes of the author of a message, the topology of a thread, and the thread context of a message. The rest of this paper is organized as follows: in section 2, we explain unique properties of newsgroup messages and search problems incurred. In section 3, we discuss other relevant research. Then, we explain our detailed experimental methodology in section 4. Results of these experiments are reported in section 5. Finally, we summarize our contributions and discuss future research directions in section 6.

2. PROBLEM CONTEXT
In this section, we will explain the unique thread tree structure of newsgroup messages, analyze user behavior in newsgroups and present some statistics of the message collection used in this research.

2.1 Thread Tree Structure of Newsgroup Messages
Different from the structure of other web-based information content such as hyperlinks within web pages, newsgroup messages form a unique structure: the **thread tree** structure. The root of the thread tree is the first message posted by someone seeking an answer to his/her question or someone who wants to initialize a discussion regarding a specific topic, and then the thread tree expands as other people reply to this message and continue the discussion. Figure 1 is an example of a newsgroup message thread tree. A newsgroup collection can be regarded as a collection of thread trees, and there is no explicit semantic relationship between different thread trees.

Previous research (e.g., [4] [11] [31]) has explored different ways to take advantage of document tree structure (especially the Document Object Model (DOM) structure of HTML documents) to improve the effectiveness of web search. However, the thread

SIGIR'04, July 25–29, 2004, Sheffield, South Yorkshire, UK.

tree structure of newsgroup messages is radically different from the DOM structure because:

- DOM tree structure represents the structural/ functional/ visual relationship of different components with an html page. The newsgroup thread tree structure represents the semantic relationships of messages within a thread tree.
- The edge of the DOM structure carries explicit relationships (represented by the html tag) of different parts of a web page. The relationships represented by edges of the message thread trees are implicit.
- Newsgroup threads contain an interesting form of lexical redundancy, which is atypical of a web page. For example, a posting often restates/clarifies/refutes an earlier posting contained in the same thread.

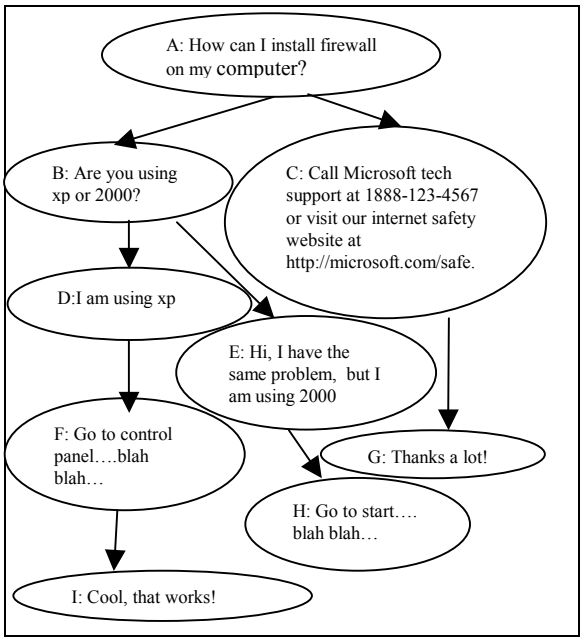

Figure 1 An example of newsgroup message thread tree

It is not difficult to notice that the content and structure of newsgroup messages are similar to email messages, which also contain semantic relationships between messages. However, searching on newsgroup messages is also different from searching on e-mail messages, because of:

Different Purpose: Users will most likely utilize a newsgroup collection as a knowledge base, and search for answers to their specific information need. However, users may regard search on emails as an advanced navigation tool, and use it to locate emails from a specific sender (perhaps restricted to a specific time) or contain specific keywords.

Different Content Property: Unlike public newsgroup messages, which are aimed at providing useful information to the whole community, email collections are private collections and are not likely to be shared with other people. User can only search on their own email collection. This also limits the size of the email collection typically searched.

Different Content Familiarity: People are familiar (or at least have a basic idea) with the content and authors of their own email collection. Searching one's own email collection is searching on known information. However, people have little idea of the contents and authors of the newsgroup messages. Searching on newsgroup message is searching on unknown content.

2.2 User Behavior in Newsgroups

Understanding user behavior in newsgroups will help to detect common user patterns and thus help us find ways to improve the search effectiveness. Some people visit a newsgroup to ask questions, others come to a newsgroup to answer other people's questions or discuss some topics. This leads to two different kinds of behavior: **Information Seeking** and **Message Posting**.

Information Seeking Behavior

There are two different ways for a user to find information via a newsgroup. They can either post a new question on the newsgroup or submit a query to the search engine. It is very important to understand the differences between the two methods.

The user-posted questions in newsgroups are usually long. (more than 20 words). They contain the very detailed situation that the user is facing. On the other hand, the queries submitted to a newsgroup search engine are usually very short, normally, 2-3 words. The information need carried by these search queries is much less detailed compared to the user posted questions.

A user typically posts a question to a newsgroup because he thinks the question is unique or he wants a very detailed answer to the question. A user submits a query to a newsgroup search engine because he believes somebody might already have answered the same question or a very similar question in the newsgroup. No matter whether a user posts a new question to the newsgroup or submits a query to the search engine, he is expecting other people's answers to his question, not other people's similar question messages.

Message Posting Behavior

Not all the messages in a newsgroup have equal content quality; some messages contain more useful information than others. Is there any way to identify high quality messages? After a close examination of a set of newsgroup message thread trees, we found the following to be the most prevalent relationship types between messages in a thread:

1. Question relationship: a user may not be clear about the information in the previous message(s) and as a result, raises more questions in the replied message. This type of relationship is a very good indication of a shift in topic.
2. Answer relationship: Current message answers the question of the previous message(s). This type of relationship may also indicate high quality information to user's search query.
3. Agreement/Amendment relationship: In the replied message, user expresses their agreement or adds amendment to the information presented in previous message(s). Sometimes this kind of message may also contain information that can be used to answer search queries.
4. Disagreement/Argument relationship: In the replied message, user expresses their disagreement or argument to information presented in previous message(s). Sometimes the replied message may contain evidence that can be used to answer search queries.
5. Courtesy relationship: "Thank you", "You're welcome" messages. These might be cues to predict the correctness/quality of previous messages, although they may not contain any useful information themselves.

Relationship types 2-4, are more likely to lead to high quality content that can be used to answer search queries. In this research we try to develop ways to automatically combine message context in each thread tree and return the best candidate messages in

response to a user's search query. For example, if a user's query is "install firewall", we hope to return message C, F, H to the user in Figure 1, even though some of these messages do not contain any search keywords.

2.3 Collection Statistics

This research is based on a newsgroup collection gathered from the Microsoft.Public.* subhierarchy of Usenet. The collection we used contains 973,948 messages, which form 349,953 message thread trees. The average number of messages per tree is 2.78. The physical size of the collection is 806 Megabytes.

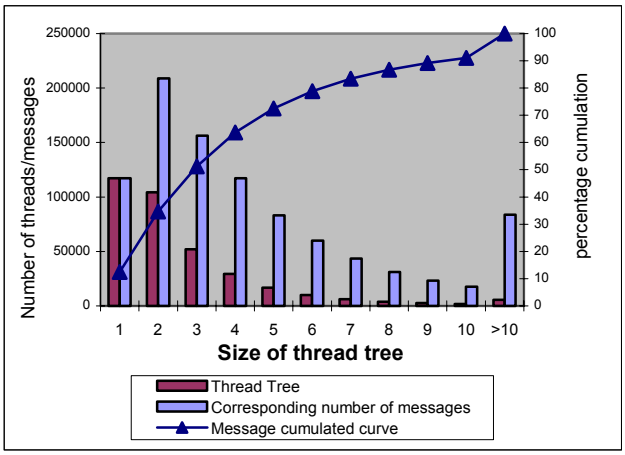

Figure 2 Thread size statistics

Figure 2 shows the thread tree size distribution in the collection. We see from this figure that the messages are not evenly distributed across different size thread trees. More specifically, 2/3 of the messages appear in the top 1/3 largest thread trees, where the size of the thread tree (number of messages in the tree) is equal or greater than 3.

This observation is important because it reveals the truth that a significant portion of messages in the newsgroup collection reside in richer (size ≥ 3) thread tree structures. And it is possible to improve the effectiveness of newsgroup search as a whole by analyzing those richer newsgroup thread tree structures.

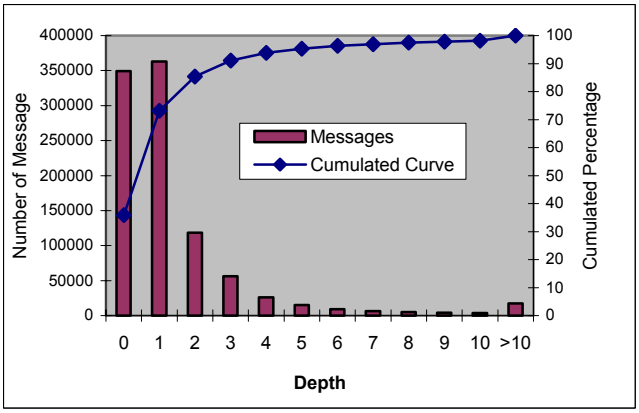

Figure 3 Message depth statistics

Figure 3 shows the message depth distribution. It is interesting to see that although over 2/3 of the messages are in thread trees with size ≥ 3, most messages in the collections reside in very shallow depth of a thread tree. (E.g. over 73% of messages are within 1 level from the root, and over 85% of the messages are within 2 levels from the root). Even after we discard all the root messages (assuming they are all question messages), we still find that over 75% of the remaining messages reside within 2 levels from the root message, and over 86% within 3 levels. The message-depth curve is highly skewed.

3. RELATED WORK

Our work on newsgroup search has roots in two research areas: structured information retrieval and data fusion in information retrieval.

3.1 Structured Information Retrieval

Our research entails analyzing the structure of message thread trees to identify/combine the messages that can best answer a user's search query. Identifying the best content segment within a document is a well-studied topic in information retrieval.

Moffat, et, al. [21][31] and Callan [1] did some early work by partitioning documents into disjoint segments of roughly equal length, and found that retrieval on passages with 150-300 words was significantly more effective than full document retrieval. Salton et. al [23] used a two-pass method. First, they searched a query against the full document to obtain a top document list. Then, passage level similarity was used to reorder this list. They found this approach improved retrieval effectiveness as well. Fixed length window or natural boundary separation technology discussed above has proven to be robust and effective for document retrieval involving fairly long documents.

Hearst et. al.[13] considered documents as sequences of locally concentrated discussions and used a method to automatically group small paragraphs into topically related segments. Her approach retrieves the top segments against the query and sums the scores of all segments from the same document. This approach was shown to be better than both document and single passage retrieval. Her work is related to ours because newsgroup thread trees can also be considered as a set of locally concentrated discussions, but different to high quality traditional literature used in Hearst's research, newsgroup message threads contain messages authored by different people and are not linear in structure. In effect, a newsgroup thread can be seen as an already grouped topically related set of messages.

In one other work, Mittendorf and Schauble [19] also suggested using inferred passage boundaries, by employing a hidden Markov model to determine passages appropriate to each query. They found that passage ranking improved retrieval effectiveness.

With the rapid growth of the World Wide Web, more research has been done to analyze the structure of web pages to improve web-search. Embley[5] uses heuristic rules to discover record boundaries within a page to assist data extraction from the web page. Chakrabarti [2] addresses the fine-grained topic distillation and dis-aggregates hubs into regions by analyzing link structure as well as intra-page text distribution. Chen [4] developed a Function-based Object Model (FOM) of web page for content understanding and adaptation. Gu [11] tries to construct a web content structure by breaking out the DOM tree and comparing similarity among the basic DOM nodes. Most recently, Yu et. al [30], proposed the use of visual cues in the web-page to detect web page content structure. Most of this work analyzes the structural/functional/visual relationship of different components of an html page. Our research focuses on analyzing the semantic relationship of different messages within a thread

tree, and we also try to combine multiple evidence sources discovered from message relationships, as well as lexical distribution in threads and author attributes, automatically.

3.2 Data Fusion in Information Retrieval

Information retrieval can be considered as using a set of known evidence (e.g. document/ query term frequency, web link structure) to predict an unknown property: the relevancy of a document to a query. Extensive research has been done to combine multiple sources of evidence to improve the effectiveness of retrieval (also referred to as the "data fusion" problem). There are two generic ways to combine evidence; the first method is to combine scores from different systems/ranking schemes. Fox [24], Fuhr [8] and Gey [10] did some early work in this area. Lee [16][17] combined results from pairs of different weighting schemes from the SMART system and found that significant improvements could be obtained by combining two different weighting schemes. Vogt and Cottrell [25] combined the retrieval scores from two systems by taking a weighted sum to improve "routing queries". The effectiveness of the combined system was compared with that of the two individual systems, as well as with the best possible performance. The result showed the combined system performed virtually identically to the better of the two systems. The other fusion method is to combine different document representations to improve search effectiveness. This method is extensively used by the participants of TREC web track in recent years. Most recently, Ogilvie and Callan [21] analyzed the conditions for successful combination of different document representations based on TREC web collection, and they found that the hypotheses for meta-search on classical text collections do not necessarily hold in web environments. Research on combining different document representations is relevant to our study, because in our work, we try to combine information from a set of messages from different parts of a thread tree to improve retrieval, and different message sets can be regarded as different information representations for the same thread tree.

Chen [3] explored a range of machine-learning methods for combining four factors in information retrieval. These methods include logistic regression, linear regression, neural networks, and the linear discriminant. He found these methods gave equally good results. Xi and Fox [27] use a decision tree to separate homepage from non-homepages in a web collection and further use logistic regression to improve the performance of the "homepage finding" search task. Lewis [18] used Support Vector Machines (SVM) in Batch Filtering and Routing tasks. Most recently, Fan et. al.,[6] used Generic Programming (GP) to automatically optimize a search engine ranking function.. Tested on the traditional TREC collection, their GP optimized ranking formula performs significantly better than that of a classical search formula. In this work, we use linear regression and Support Vector Machines to combine features extracted from the structure of newsgroup thread trees to improve the effectiveness of newsgroup search.

4. EXPERIMENTAL APPROACH

We use linear regression and Support Vector Machines to investigate the effectiveness of combining features based on thread structure, lexical distribution and author attributes to create an effective ranking function for newsgroup search. The experimental steps are:

1. Building up a baseline Okapi search engine;
2. Rating a set of <query, message> pairs by human experts;
3. Extracting feature vectors for each message in our <query, message> data set;
4. Building a ranking function using Linear Regression (LR) and Support Vector Machines (SVM).

We explain each of the steps in detail below:

4.1 Building the baseline search engine

The baseline search system was developed using the data collection introduced in 2.3. The collection contains 978,943 messages from the Microsoft.public.* usenet subhierarchy.

We chose Okapi BM 2500 [22] as our default weighting function in the baseline search engine. The BM 2500 weighting function is:

$$\sum_{T \in Q} w^{(1)} \frac{(k_1+1)tf}{(k_2+tf)} \frac{(k_3+1)qtf}{(k_3+qtf)} \quad (1)$$

where Q is a query containing key terms T, tf is the frequency of occurrence of the term within a message; qtf is the frequency of the term within the query, from which Q was derived, $w^{(1)}$ is the Robertson/Spark Jones weight of T in Q. It is calculated by:

$$w^{(1)} = \log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} \quad (2)$$

where, N is the number of messages in the collection, n is the number of messages containing the term, R is the number of messages relevant to a specific topic, and r is the number of relevant documents containing the term. In (1), k_2 is calculated by: $k_1((1-b)+b \times dl/avdl)$, where dl and $avdl$ denote the document length and the average message length measured in words. In this experiment, we set $k_1 = 1.2$, $k_3 = 1000$, $b = 0.75$.

In the baseline search system, we use a naïve stemming method (e.g., only truncate the words ending in "ing" and "s"). We use a list of approximately 500 stopwords.

4.2 Rating the <query, message> pairs

Human experts were hired to manually examine and identify the relevancy of top messages returned by the baseline Okapi search engine for a set of queries. The annotated <query, message> pairs are then used to train and evaluate our ranking algorithms.

We use 343 sample queries drawn from the Microsoft Help & Support website search log to search against the newsgroup message collection. For each of the sample queries, at most the top 25 messages returned by the baseline search engine were kept. Thus, we obtained a collection of 5552 <query, message> pairs. Two domain experts on Microsoft product knowledge were hired to judge the relevance of these pairs.

When rating each of the <query, message> pairs, each domain expert was allowed to choose one of the following options and associated score: relevant (3.0); partially relevant (2.0), non-relevant (1.0), can not tell (no score associated). Only <query, message> pairs receiving a score of no less than 2.5 on average[1] were regarded as relevant pairs. All the others pairs were regarded as non-relevant pairs.

In order to prevent the domain experts from misunderstanding the queries, we gave a short description for each of the 343 sample queries, in a way very similar to the

[1] i.e. one human labeled a pair as relevant and one labeled it as partially relevant.

<desc> part of queries used in TREC ad-hoc tasks [12]. Table 1 shows a few sample queries with corresponding descriptions.

Table 1. Sample queries and their descriptions

Queries	Descriptions
Norton Antivirus 2001	What is this software? How can I get/install this software? Info of specific virus that can be fixed by this software will not be relevant.
Boot disk	What is this, how do I create one? Problems that can be fixed by using this will not be relevant.
Dual Booting	What is this? How can I do this? Specific problem caused/solved by dual booting will not be relevant.

After the rating process, we found out that 1008 <query, document> pairs (approximately 18% of all the pairs) were relevant.

4.3 Feature Extraction

Two kinds of features are collected for each of the messages in the thread tree; they are features from the thread tree, and features from the author of the message.

First, we explain features extracted from the message thread tree. These features are obtained by combining features from 2 domains: ***tree structure***, ***ranking function***.

Tree Structure: Since some messages may include text from previous message(s), we first filter out all the previous messages enclosed and only leave the core part of the message to prevent the system from generating duplicate message contents in the subsequent steps. Then, we link each message to its previous message and its following messages to rebuild the message thread tree. For each of the messages in the thread tree, we extract content of another message or a combination of other messages in the same thread tree according to the relationships described in Table 2.

Table 2. Content features from the thread tree structure

Name	Explanation
Message	The current message core.
Title	The title of the current message.
Root	The root message of the current message.
Parent	The current message's direct parent message. (Null, if current message is root).
Ancestor	All the ancestor messages of the current message, up to the root message.
Thread	The current message, and all its ancestor messages, up to the root message.
Non-root	Thread, without the root message.
Children	All the direct child messages of the current message
Descendant	All the descendant messages of the current message.

Ranking Functions*:* for each of the content features extracted from the thread tree structure, we calculate 3 scores according to different ranking functions as explained in Table 3.

Table 3. Different ranking functions

Ranking function	Explanation
Okapi Score	The standard BM2500 used in the baseline system
Binary Score	1/0 if a query term appears/does not appear in a message. The final score is the sum of term scores for all the term in the query. This method is reported to be good for short query search [28].
TotalTF Score	Total number of keywords occurring in the message

With 9 content features selected from the thread tree structure, and 3 different ranking functions, we have collected 27 attribute scores for each of the messages. We also calculate 3 different ranking scores for the original message (without pruning out the enclosed previous messages), and that gives us a total of 30 query-specific features extracted from each message. Other query independent features collected from the thread tree are shown in Table 4:

Table 4. Content independent features from thread tree

Name	Explanation
IsRoot	Binary score indicating whether the message is a root message or not.
Generation	Which generation current message occurs in.
NumberofChildren	Total number of direct children the current message has.
TotalDescendant	Total number of descendants that the current message has.
DescendantDepth	The biggest depth of the current message's descendant messages.
TotalLeaf	The total number of leaf messages in the descendants of the current message.

Beside the features extracted from the message thread tree, features from the author of each of the messages also are collected using the Microsoft Research NETSCAN project. In earlier work, Fiore et. al, [7] showed that some of these features correlated with how likely a user would like to read another posting from an author. In this work, we tried to find whether such features could help our ranking function by providing a query-independent author-quality assessment (akin to Pagerank in Web search). These features are explained in Table 5.

Table 5. Features from Author's matrix

Name	Explanation
Posts	Number of messages an author posts during a period of time.
Replies	Number of reply messages an author posts during a period of time.
Reponses	How many responses the author gets.
AverageLineCount	Average length of his/her posting.
DaysPresent	Days he/she posts to Usenet.
ThreadCount	How many threads he/she is involved in.
Starts	How many threads he/she starts.
Barren	How many of his/her messages get no replies
NewsGropupCount	How many newsgroup communities he/she is active in.

We have collected 45 features for each of the 5552 <query, message> pairs to train and test our ranking functions.

4.4 Combining multiple features

In this work we use Linear Regression (LR) and Support Vector Machines (SVM) to train ranking functions using the features described above.

The most common way of combining multiple sources of evidence in information retrieval is to take a weighted sum of the scores of different evidence: $S = \sum w_i s_i$. Where s_i are scores from different sources, w_i is the correspondent weights and S is the final score for the document. In Linear Regression analysis the dependent variable S is a boolean factor indicating whether the message is relevant (1) to a query or not (0). The independent variables are 45 features described in previous section. Regression analysis can be used to find a set of weights w_i, which best predicts the dependent variable (i.e. the probability of a message relevant to a query).

Support Vector Machiness (SVM) are also linear functions of the form $f(x) = \boldsymbol{w} \bullet \boldsymbol{x} + b$, where $\boldsymbol{w} \bullet \boldsymbol{x}$ is the inner product between the weight vector $\boldsymbol{w}$ and the input vector $\boldsymbol{x}$. the SVM are mostly used as classifiers to classify class 1 $f(x)>0$ from class -1 $f(x)\leq 0$ The main idea of SVM is to select a hyper-plane that separates the positive and negative examples while maximizing the minimum margin, which is defined as $y_i f(x_i)$ for x_i, where $y_i \in \{-1,1\}$ is the target output. This corresponds to minimizing $\boldsymbol{w} \bullet \boldsymbol{w}$ subject to $y_i (\boldsymbol{w} \bullet \boldsymbol{x}_i + b) \geq 1$ for all i. In this research we consider relevant messages are in class 1 and non-relevant messages are in class –1. Vector $\boldsymbol{x}$ consists of all the features we have extracted. The object of the SVM is to find the weight vector $\boldsymbol{w}$ that can best separate relevant messages from non-relevant messages. We utilize the classifiers as ranking functions by sorting the documents according to the weight assigned by the classifier.

5. EXPERIMENTAL RESULTS

The original motivation of this work was to build an effective ranking function for usenet searches involving queries relevant to Microsoft products. We obtained 343 random queries from Microsoft Help and Support Search Query logs. We had a collection of 973948 messages from the Microsoft.public.* subhierarchy of Usenet. We used okapi baseline to find the top 25 messages for each query. We then had people annotate these messages, giving us a set of 5,552 <query, intent, message> triples. We split the data set into training instances and test instances. The training data set contains instances from 75% of the queries and the test data set contains instances from the rest 25% of the queries. There was no overlap in queries in the training and test data.

In the case of product support search, users typically care more about rapidly finding a satisficing answer, rather than the number of relevant messages retrieved. If the first return message correctly explains to the querier how to install a firewall in Windows XP, it is irrelevant whether subsequent returned messages also correctly answer the query. For this reason, we chose *Mean Reciprocal Rank (MRR*) of the first relevant message to evaluate the models we developed using LR and SVM. *MRR* is defined as:

$$MRR = \frac{\sum_{i \in N} (1 / R_i)}{N} \qquad (3)$$

where, Ri is the rank of first correct answer for query i, and N is total number of queries.

The performance evaluation of our trained ranking functions is reported in Figure 4.

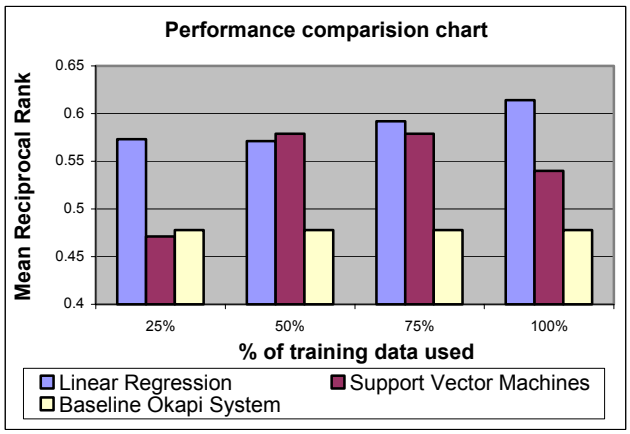

Figure 4 the evaluation for different sizes of training data

We can see from Figure 4 that ranking functions trained using both Linear Regression (LR) and Support Vector Machines (SVM) can give a significant improvement over the baseline Okapi system. The Linear Regression Model trained on the entire 4,164 <query, message> pair training set achieved a 28.5% improvement over our baseline Okapi system. We used a two-tailed t-test to compare the *MRR* of LR with that of the baseline Okapi system for the testing queries and found that this improvement is significant at $\alpha = 0.00006$. Moreover, as the training data increases, the performance of the Linear Regression trained ranking function seems to be more robust and consistent than the performance of Support Vector Machines. As the training data increases, the performance of Linear Regression model shows a steadily increasing learning curve, indicating that further improvement can likely be achieved, if we further increase the training data set. This is something we plan to investigate in the future.

The retrieval performance of the best LR Model was also compared with the best single features we collected in 4.3. Results are shown in Figure 5. We can see from the figure that the best LR Model beats the best single evidence: "the Okapi score obtained from the Non-root portion of the thread tree (OkapiNonroot)" by 13.5%. A 2-tailed T-test performed on the results of the LR Model and OkapiNonRoot show that the improvement is statistically significant at $\alpha = 0.0003$. Another way of looking at this is that approximately 45% of our improvement gain over the baseline can be accounted for by the OkapiNonRoot feature.

We can also see from the figure that most of the best single features are Okapi scores obtained from different parts of the thread tree.

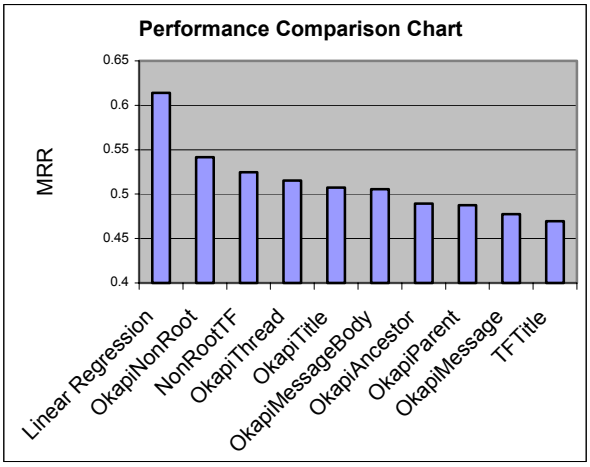

Figure 5. Performance comparison for different single features with the Linear Regression Model

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

Newsgroup search is a very important service for web communities; however, the performance of a traditional text based search engine, or a search engine for web pages, when used on newsgroups, is not adequate due to unique attributes of newsgroup message threads. In this paper, we analyzed the reasons that cause the text-based search engine to fail on newsgroup search and also analyzed the thread tree structure of newsgroup messages, then we proposed that the performance of newsgroup search can be improved by analyzing the context information of the messages within the message thread tree.

After extracting a set of features from the related messages in the same thread tree and author information for each of the messages in the newsgroup, we used Linear Regression (LR) and Support Vector Machines (SVM) to combine these features to create a ranking function. Our experiments achieved a 28.5% improvement over the baseline Okapi system (in terms of MRR), which confirmed our assumption that the performance of newsgroup search can be enhanced by incorporating features relevant to the newsgroup domain, such as thread context information. We were surprised to learn that author features (which we expected would serve to indicate message quality much like PageRank does for Web Pages), did not appear to play a significant role in our ranking functions. Furthermore, we found that with Linear Regression, the performance is proportional to the amount of training data used. This suggests an even larger scale human annotation on test data might achieve higher performance.

6.2 Future Work

Although significant improvement on the performance of newsgroup search has been achieved by combining multiple features from the context of the message in the message thread tree, there are still more ways to improve the performance of newsgroup search left unexplored.

One method is to try to automatically detect the semantic relationship of the message to its previous messages, or in other words, find the relationship information carried by the edges on the newsgroup thread tree. Although in section 2.2 we detected 5 kinds of such relationships, in this research, we did not try any methods to automatically identify these relationships and combine them together with the content based features to improve the search performance. The usefulness of these relationships for search is still to be found.

The focus of this project was to create a message ranking function for newsgroup search to help people more readily find information in newsgroups relevant to Microsoft help and support related queries. In the future, we plan to study to what extent our results and conclusions are applicable to all newsgroup search in general.

7. ACKNOWLEDGEMENT

The authors would thank Dr. John Platt for his SMOX Support Vector Machine tool, Dr. David M. Chicking for his Linear Regression tool, and Dr. Marc Smith for his NetScan web social data mining tool kit. The authors would also thank Dr. Susan Dumais for her valuable suggestions.

8. REFERENCES

[1] J.P. Callan, “Passage-level evidence in document retrieval.” In Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Dublin, Ireland, pp. 302-309. 1994.

[2] S. Chakrabarti, M. Joshi, and V. Tawde, “Enhanced topic distillation using text, markup tags, and hyperlinks.” *In Proceedings of the 24th annual international ACM SIGIR Conference on Research and development in information retrieval*, New Orleans, Louisiana, pp. 208-216, 2001.

[3] A. Chen. “A comparison of regression, neural net, and pattern recognition approaches to IR,” *In Proceedings of the 1998 ACM 7th International Conference on Information and Knowledge Management (CIKM ’98),* Bethesda, Maryland, pp. 140-147, 1998.

[4] J. Chen, B. Zhou, J. Shi, H. Zhang, and Q. Wu, “Function-Based Object Model Towards Website Adaptation,” *In Proceedings of the 10th International World Wide Web Conference*. Hong Kong, China, pp. 601-610, 2001.

[5] D.W. Embley, Y. Jiang, and Y.-K Ng, “Record-boundary discovery in Web documents,” *In Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, Philadelphia PA, pp. 467-478, 1999.

[6] W. Fan, M.D. Gordon, P. Pathak, “A generic ranking function discovery framework by genetic programming for information retrieval,” *Information Processing and Management*, in press, 2004.

[7] Fiore, Andrew, Scott Lee Teirnan, Marc Smith. "Observed Behavior and Perceived Value of Authors in Usenet Newsgroups: Bridging the Gap", *In Proceedings of the CHI 2002 Conference on Human Factors in Computing Systems*, Minneapolis, Minnesota, pp. 323-330, 2002.

[8] N. Fuhr. “Integration of Probabilistic Fact and Text Retrieval.” *In Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Copenhagen, Denmark, pp. 211-222, 1992.

[9] S. Fujita. “More Reflections on "Aboutness"”. TREC-2001 Evaluation Experiments at Justsystem. *In Proceedings of the*

Tenth Text Retrieval Conference (TREC 2001). Gaithersburg, Maryland, NIST Special Publication 500-250. 2002.

[10] F.C. Gey, A. Chen, J. He, and J. Meggs. "Logistic regression at TREC4: Probabilistic retrieval from full text document collections." *In Proceedings of the Fourth Text Retrieval Conference (TREC 4)*. Gaithersburg, Maryland, NIST Special Publication 500-236. 1996.

[11] X. Gu, J. Chen, W.-Y., Ma, and G. Chen, "Visual Based Content Understanding towards Web Adaptation*," In Proceedings of Second International Conference on Adaptive Hypermedia and Adaptive Web-based Systems (AH2002),* Spain, pp. 29-31, 2002.

[12] D.K. Harman, "Overview of the Fourth Text Retrieval Conference (TREC-4)," *In Proceedings of the Fourth Text Retrieval Conference (TREC-4),* Gaithersburg, Maryland, NIST Special Publication 500-236, pp. 1-23, 1995.

[13] M.A. Hearst, and C, Plaunt "Subtopic structuring for full-length document access." *In Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, Pennsylvania, pp. 59-68, 1993.

[14] G. Kazai, M. Lalmas and T. Roelleke. "A Model for the Representation and Focused Retrieval of Structured Documents based on Fuzzy Aggregation", *In Proceedings of the 8th International Symposium on String Processing and Information Retrieval*, Laguna de San Rafael, Chile, pp. 123-135, 2001.

[15] Kaszkiel M., & Zobel, J. (1997). "Passage retrieval revisited." *In Proceedings of the Twentieth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Philadelphia, Pennsylvania, pp. 178-185. 1997.

[16] J.H. Lee. "Combining multiple evidence from different properties of weighting schemes." *In Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Seattle, Washington, pp. 180-188. 1995.

[17] J.H. Lee. Analyses of Multiple Evidence Combination. *In Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Philadelphia, Pennsylvania, pp. 267–276. 1997.

[18] D. Lewis "Applying Support Vector Machines to the TREC-2001 Batch Filtering and Routing Tasks". *In Proceedings of the Tenth Text Retrieval Conference (TREC 2001).* Gaithersburg, Maryland, NIST Special Publication 500-250. 2002.

[19] E. Mittendorf, and P. Schauble, "Document and passage retrieval based on hidden Markov models*." In Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Dublin, Ireland, pp. 318-327. 1994.

[20] A. Moffat, R. Sack-Davis, R. Wilkinson, and Zobel, J. "Retrieval of Partial Document*." In Proceedings of the Second Text Retrieval Conference (TREC-2),* pp.181-190. NIST Special Publication pp. 500-215, 1994.

[21] P. Ogilvie, J. Callan: "Combining document representations for known-item search." *In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Toronto, Canada pp.143-150, 2003.

[22] S.E. Robertson, "Overview of the Okapi Projects, *Journal of Documentation*, Vol. 53, No.1, pp. 3-7, 1997.

[23] G. Salton, J. Allan, & C. Buckley, C. "Approaches to passage retrieval in full text information systems." *In Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, Pennsylvania, pp. 49-58, 1993.

[24] J.A. Shaw & E.A. Fox, "Combination of multiple searches", *In Proceedings of the 3rd Text Retrieval Conference (TREC-3)*. Gaithersburg, Maryland: NIST Special Publication 500-250, pp.105-107, 1995.

[25] C.C. Vogt and G.W. Cottrell. "Fusion via linear combination for the routing problem*". In Proceedings of the Sixth Text Retrieval Conference (TREC 2001).* NIST Special Publication 500-250. 1998.

[26] R. Wilkinson, "Effective retrieval of structured documents." *In Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Dublin, Ireland, pp. 311-317. 1994.

[27] W. Xi and E. A. Fox. "Machine Learning Approach for Homepage Finding task". *In Proceedings of the Tenth Text Retrieval Conference (TREC 2001).)*. Gaithersburg, Maryland, NIST Special Publication 500-250. 2002.

[28] W. Xi, "Combining Multiple Source of Evidence for Information Retrieval," *Master Thesis*, Nanyang Technological University, Singapore, 2000.

[29] J. Xu, and W.B. Croft. "Query expansion using local and global document analysis." *In Proceeding of ACM-SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland, pp.4-11, 1996.

[30] S. Yu, D. Cai, J-R Wen and W-Y. Ma, "Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation", *In Proceeding of the Twelfth World Wide Web conference (WWW 2003),* Budapest, Hungary, pp.11-18, 2003.

[31] J. Zobel, A. Moffat, R. Wilkinson, and R. Sack-Davis, "Efficient retrieval of partial documents." *Information Processing & Management*, 31(3), 1995.