Author-Oriented Model for Information Retrieval

4th May 2004

William Lee
University of Illinois at Urbana-Campaign
wwlee1@uiuc.edu

CheungXiang Zhai
University of Illinois at Urbana-Champaign
czhai@cs.uiuc.edu

Abstract

This paper proposes a new model for information retrieval through the use of an author's pass information. This model assumes that if an author writes more about a particular topic, then the document that he/she have written should be more insightful than the other similarly relevant documents. Therefore, those documents written by experienced authors should be ranked higher. This paper describes an application of this model to retrieving messages from a mailing list or a newsgroup environment. In particular, we have derived an architecture named CEES, or Conversation Extraction and Evaluation Service, to archive, index, and retrieve documents from a mailing list or newsgroup. We have also conduct a preliminary retrieval performance evaluation based on a set of tagged mailing list messages.

1 Introduction

Many useful technical conversations take place in newsgroups and mailing lists every day. Even with the growth of the WWW and web-based forums, Usenet and mailing list still represent a large portion of the internet traffic. For instance, most open source software projects have one or more mailing lists. Various issues are discussed in the forums varying from simple questions on product installation to expert configuration for large scale load balancing.

As an example, the Apache Tomcat (a popular Java web application server) user mailing list has on average around 13MB of mail traffic each month from March 2000 to March 2004. There are 37587 messages exchanged in that mailing list alone in 2003. This implies a subscriber to that mailing list on average receive more than 100 messages a day! For a new subscriber, picking out relevant information from this traffic is a challenging task, even though many mailing lists have searchable archives dated several years back. However, a casual survey on several mailing lists results in many repeated messages asking the poster to first search the mailing list archive. This shows that the ineffectiveness for the existing search system.

Although retrieving documents from the Usenet and mailing resembles retrieving a plain text or web document, mailing list messages have some unique features that can be used to potentially improve retrieval performance. First of all, some structured information within the document is easily obtained through the Subject and From/To headers. Retrieval systems can also derive inter-message relationship using the In-Reply-To and References header. Rather than considering messages as separate documents, a model that utilizes the mailing list's background information may have desirable properties to better satisfy information need. Using such information, one may expect to improve the retrieval performance.

In a mailing list or newsgroup, people can be generally divided into two groups: "seekers" that look for information, and "providers" that has the expertise to answers those questions. This separation is, of course, not concrete, since many providers are seekers as well. One would expect, however, that the providers that have posted multiple times before will have more relevant messages that can satisfy the seekers. For example, in the Tomcat mailing list, one would expect a Tomcat developer will have more relevant posts than a newcomer that recently joins the mailing list. Those messages posted by the developer should deserve a better ranking.

For new subscribers to the mailing list or newsgroup, even though they knows what they want when searching for the mailing list archive, it is inconceivable to assume that they can evaluate the quality of the documents retrieved by the system. For one thing, the retrieval system can potentially return many relevant messages, since their questions have probably been asked many times before. Since new subscribers do not generally know the other subscribers in a public mailing list, all they may see is a collection of varying responses to their questions. However, they have no way to know the credibility of the authors behind those messages. Our proposed model attempts to fill in the background knowledge for the user and incorporate this information in the retrieval results.

2 Background

Not much work on information retrieval tried to evaluate the retrieval performance using background information in a newsgroup or a mailing list context. There are several papers from the user interface domain that have experimented with alternative ways to visualize the conversation structure for better understanding of the newsgroup structure [1, 6]. In particular, one of Andrew's "piano roll" display includes a ranked list of authors who contributes the most number of posts in the thread. Andrew believes that this component can help users identify the more relevant posts. However, it is not known how much of this information can potentially affect the information retrieval performance.

Fundamentally, most information retrieval methods would apply to the mailing list context without much modification. In particular, since email messages in general has more defined structure, we can potentially leverage the work done on structured information retrieval [3, 4, 5, 7] and various smoothing strategies [8, 2].

The work done in this paper has not incorporated the more sophisticated methods such as the mixture model and parameter estimation in more recent information retrieval works. More advanced techniques should be examined and experimented in the future.

3 A Model for Author-Oriented Information Retrieval

Our new model divided a newsgroup message into 3 parts, the subject, the body, and the document model represented by the author of the message. They are all considered as part of the overall document model for retrieval. The basic retrieval method uses the Kullback-Leibler divergence retrieval model described by Zhai and Lafferty in [9]. In such model, a document is ranked by its similarity of distribution between the query and the document model. In particular, given two distribution functions p and q, the KL divergence, or relatively entropy, $D(p||q)$, is defined as:

$$D(p||q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

In a generative model a query q is generated by the query mode θ_Q and the document d is generated by the document model θ_D. Their best estimated models are $\hat{\theta_Q}$ and $\hat{\theta_D}$ respectively. The score of

the document can be expressed as their KL divergence:

$$D(\hat{\theta_Q}||\hat{\theta_D}) = \sum_w p(w|\hat{\theta_Q}) \log \frac{p(w|\hat{\theta_Q})}{p(w|\hat{\theta_D})}$$

Essentially, we want the most probable query model that generated the query and the document model that generates the document to have minimum divergence. Therefore, we take the negative divergence [9] instead. The formula becomes:

$$-D(\hat{\theta_Q}||\hat{\theta_D}) = \sum_w p(w|\hat{\theta_Q}) \log p(w|\hat{\theta_D}) - \sum_w p(w|\hat{\theta_Q}) \log p(w|\hat{\theta_Q})$$

We can ignore the right-most component for the purpose of ranking document, since it describes only the entropy of the query model $\hat{\theta_Q}$. Note that:

$$p(w|\hat{\theta_D}) = \begin{cases} p_s(w|d) \, if \, w \, is \, seen \\ \alpha_D p(w|C) \, otherwise \end{cases}$$

The probability $p_s(w|D)$ is the probability that the word w is seen in document d, and $p(w|C)$ is the probability of the unseen word w in the entire collection. The parameter α_d is a weighting parameter to balance between the seen and unseen word. Given $p_s(w|D)$, we must have:

$$\alpha = \frac{1 - \sum_{w:c(w;d)>0} p_s(w|d)}{1 - \sum_{w:c(w;d)>0} p(w|C)}$$

and it can be shown that from [9]:

$$\sum_{w:c(w;d)>0} \log p(w|\hat{\theta_Q}) \frac{p_s(w|d)}{\alpha_d p(w|C)} + \log \alpha_d$$

can be used to rank the documents. Using Dirichlet prior smoothing for which:

$$p_s(w|d) = \frac{c(w;d) + \mu p(w|C)}{|d| + \mu}$$

, we can set α_d to

$$\alpha_d = \frac{\mu}{|d| + \mu}$$

where μ is the Dirichlet prior.

The first two components in our model, the body and subject, can be computed efficiently with the KL divergence method using two inverted indices. The body index and the subject index store the term to document ID (and its associated document statistics) for the body text and subject text respectively. The last component, the document model for the author A, involves the assumption that the author is represented by all the documents that he/she has written before. The author's relevance depends on the query, since different authors are experts on different topics. Essentially, we would like to get a ranking of the author based on the input query.

Suppose A_d represents the author for the document d, the ranking of the author can be captured in:

$$\begin{aligned} P(A_d|Q) &= \frac{P(Q|A_d)P(A_d)}{P(Q)} \\ &\propto P(Q|A_d)P(A_d) \end{aligned}$$

The term $P(A_d|Q)$ estimates the probability that the author that has written the document d is an expert to the query Q. For now, we can make the assumption that the $P(A_d)$ is uniform, although this statistics can also be computed by counting how often the author has posted the on the newsgroup. Therefore, we only need to use $P(Q|A_d)$ for ranking. We can compute this by using the simple unigram model:

$$P(Q|A_d) = \prod_{i=1}^{|Q|} P(q_i|A_d)$$

We can use the KL divergence method to compute $P(Q|A_d)$ as well, if we treat the author model A_d the same as the document model θ_D described in the previous context. In order to use the conventional index model, however, we need to treat the set of documents written by A_d as one combined meta document. For instance, given that we have documents $d_1, d_2, ..., d_n$ in the collection C, we would like to make a subset S_{A_d} of $d_i \in C$ such that d_i is written by A_d. We then treat S_{A_d} as one large document used for indexing. The document ID, in this case, would be an unique identifier for the author A_d. In our case we use the message poster's email address as the meta document's ID.

A heuristic way is used to combine the final score for ranking. First, each document is scored and ranked based on the KL divergence method. For the general index, we essentially acquire a ranked list of documents $G = \{(G_1, g_1), (G_2, g_2)...(G_n, g_n)\}$ such that G_i is the i^{th} highest ranked document and g_i is its associated score. We then do the similar thing for the subject index, where $S = \{(S_1, s_1), ..., (S_k s_k)\}$. Finally, we apply the same method on the author index such that $A = \{(A_1, a_1), (A_2, a_2), ..., (A_m, a_m)\}$. A_i is an individual author and a_i represents the associated score for the "insightfulness" of the author base on the query.

After we have calculated G, A, and S using the KL divergence method, we discard the scores g_i, a_i, and s_i for all i. Instead, we use the ranking of the document d and the ranking of A_d to generate a new score for d. In other words, if for a document d such that $d = G_l$, $d = S_r$, and $A_d = A_t$(d is ranked l^{th} for the the body, r^{th} for the subject, and its author is ranked t^{th} for the given query), the new score of d is:

$$s(d) = \alpha_1 weight(d, G) + \alpha_2 weight(d, S) + \alpha_3 weight(A_d, A)$$

where:

$$weight(d, G) = \begin{cases} \frac{1}{l} \text{ if } d \text{ is in } G \\ \frac{1}{|G|} \text{ otherwise} \end{cases}$$

$$weight(d, S) = \begin{cases} \frac{1}{r} \text{ if } d \text{ is in } S \\ \frac{1}{|S|} \text{ otherwise} \end{cases}$$

$$weight(A_d, A) = \begin{cases} \frac{1}{t} \text{ if } A_d \text{ is in } A \\ \frac{1}{|A|} \text{ otherwise} \end{cases}$$

The $\alpha_1, \alpha_2, \alpha_3$ are mass constants associated with each component. Different values are set in our experiment to show varying retrieval performance.

4 Conversation Extraction and Evaluation Service

The high level goal for CEES (Conversation Extraction and Evaluation Service) is to help user search for answers to their questions better. This capability is especially important in a technical

mailing list, where frequently asked questions (FAQ) are being repeatedly asked by many newcomers everyday.

In a nutshell, CEES serves three primary functions:

1. Collection of messages
2. Archiving the messages
3. Extract, evaluate, and retrieve the stored messages

From Figure 1 we can see how CEES collect the messages and archive them in the database. There are 4 main components of the system:

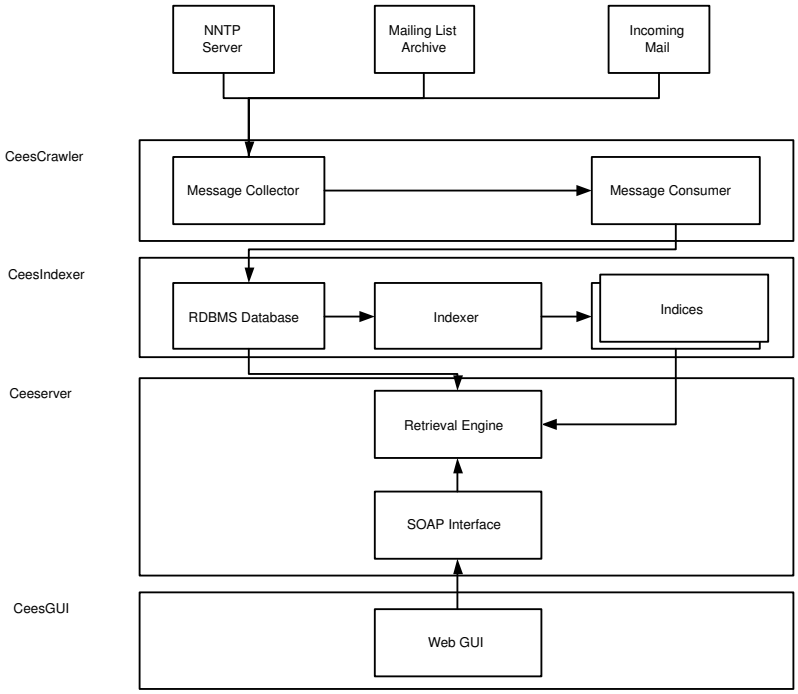

Figure 1: High-Level CEES Architecture

1. CeesCrawler consists of two main subparts:
   (a) MessageCollector collects the messages from either a NNTP server or a mailing list archive file in the standard Unix mbox format. Future implementation can potentially feed in incoming mail from an SMTP (Simple Mail Transfer Protocol) server.
   (b) MessageConsumer will take input from the MessageCollector and insert them into the RDBMS. The back-end database we used is PostgreSQL.
2. CeesIndexer reads from the messages stored in the RDBMS and index the messages into body, subject, and the author index. The indexer creates the body and subject indices in one pass. For the author index, the indexer queries the SQL database to generate the meta document for each author. It then index the meta document with the author email as the meta document's ID.

3. The Ceeserver consists of a retrieval engine and the SOAP (Simple Object Access Protocol) interface that serves as the external interface to CEES. The engine takes in user queries, performs the necessary procedures for retrieval, and returns the results through SOAP.

4. The front-end web interface uses the Ceeserver with SOAP calls and display the results to the user through HTTP/HTML.

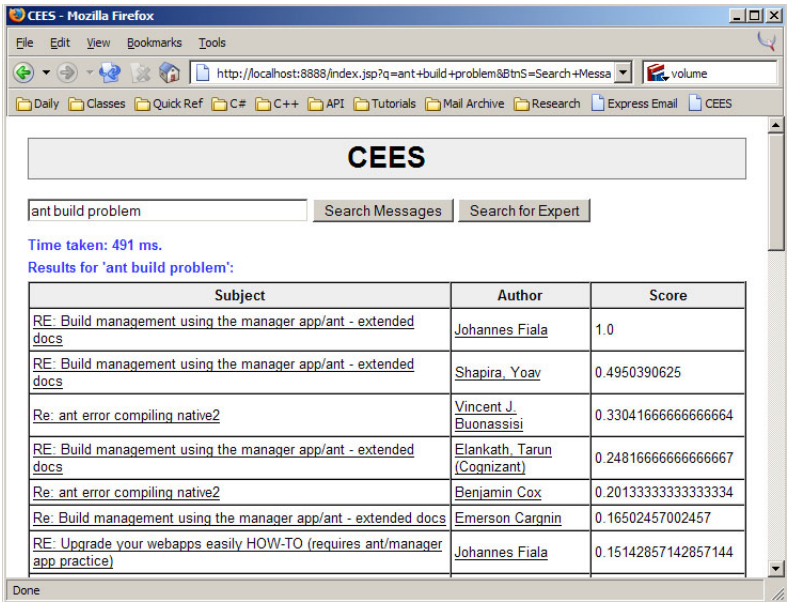

Figure 2: CEES Interface

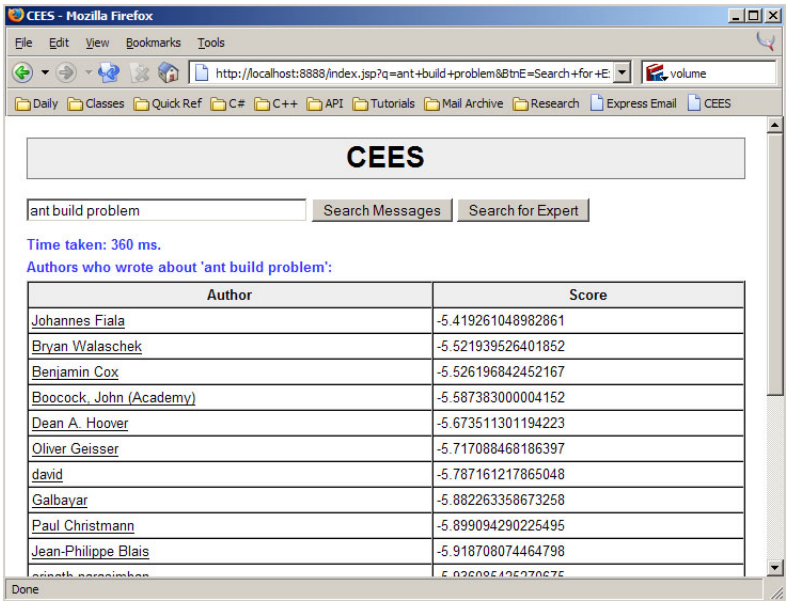

Figure 3: Searching for Experts

One can see the retrieval interface in Figure 2 and Figure 3. As a side effect of the author index, user can actually look for experts in the group that "knows" the most about the query. This is done by clicking on the "Search for Expert" button. The results you see in Figure 3 is simply a ranking of the author index A in the previous section. The users can potentially pay more attention to the people who knows the most about the query, or to email the author questions directly.

5 Evaluation and Results

Since there is no tagged data that we can use for evaluating the performance of the CEES system, a reference set is constructed from the Tomcat (http://jakarta.apache.org/tomcat) user mailing list. In particular, an archive for the month of May, 2003 is taken for this purpose. It contains 3719 messages from 772 authors. Twenty-three topics are taken from the messages, queries are constructed based on the original questions asked by real users from the mailing list.

An control set uses the body index and the KL divergence retrieval model. The Dirichlet prior μ is set to 2000 and 1000 documents are retrieved for each query. No pseudo feedback for the query is used. For the first run, α_2 is set to 0 and we vary the parameter for α_3. The result is shown in Figure 4 and Figure 5.

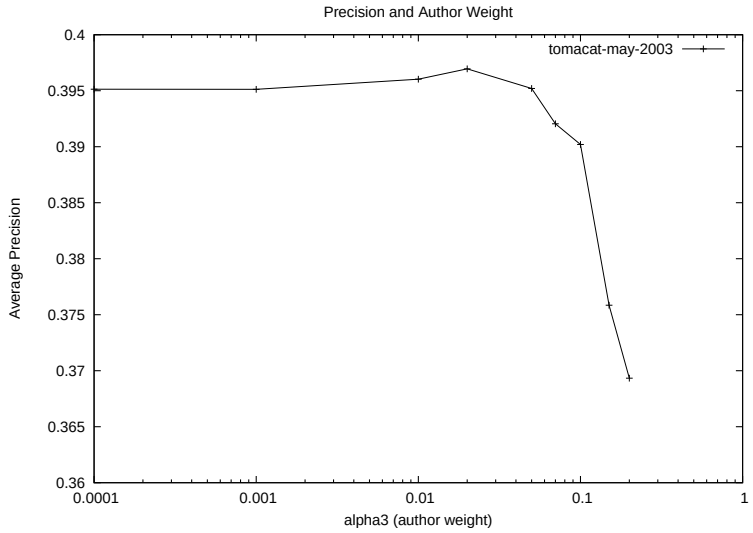

Figure 4: Performance of CEES

If we set α_3 to 0 and vary the subject constant α_2, we get the performance in Figure 6. Essentially, the performance peeks when α_2 is set to 0.05. It then rises again when α_2 is set to 0.2. The precision and recall in Figure 7 also shows that there is not much of a performance gain when altering the subject weight.

After all, there is not much of a difference between the baseline model and our new model. There is a minor improvement on the average precision if α_3 is set to 0.02. The performance degrade sharply for we overweighted the author importance if α_3 is set to anything above 0.1. This shows that the heuristic method we used is probably not ideal. We expect a different method to combine the three indices and automatic tuning of the parameters will improve the retrieval performance.

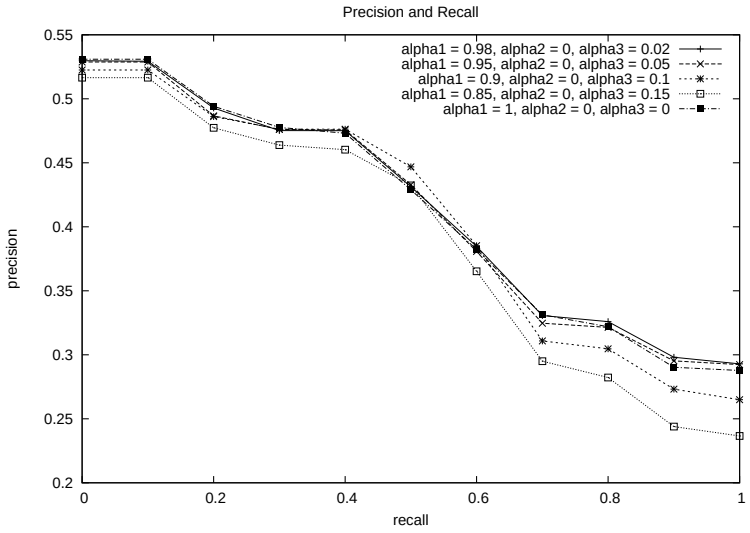

Figure 5: Precision and Recall for Author Weight

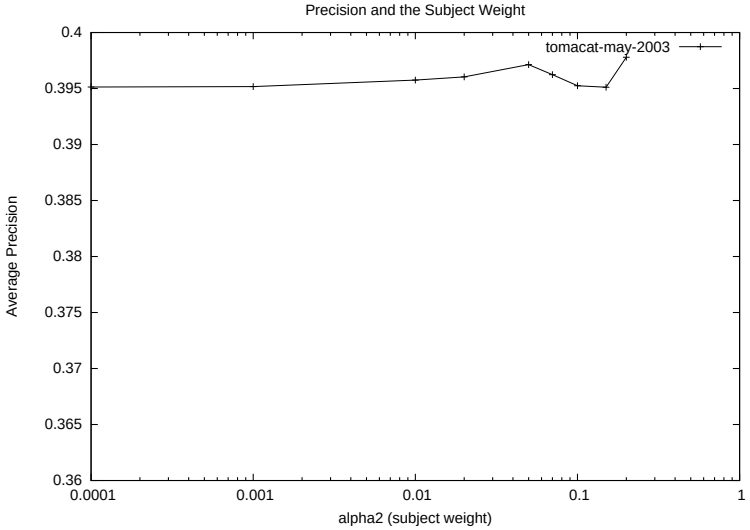

Figure 6: CEES Performance Using the Subject Parameter

6 Future work

There are much to be done in order to really see whether the author-oriented model can significantly improves the insightfulness of the retrieval.

First of all, the current way to combine the score does not take advantage of the value of the score. It's also very computationally intensive since it needs find each document in two other ranked list in order to calculate the score. A better way is to use the method suggested by Ogilvie [5]. Ogilvie proposes a method to combine various representations of the document using a variation of

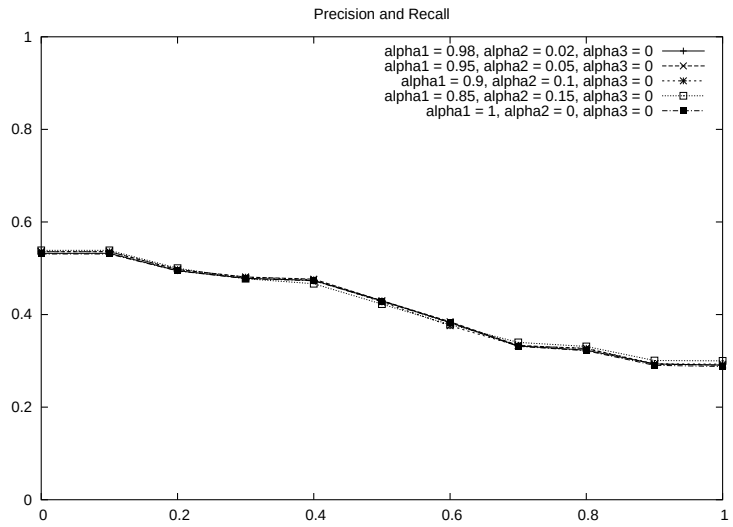

Figure 7: Precision and Recall for Subject Weight

Dirichlet prior. In particular, suppose $\{\theta_{D(1)}, \theta_{D(2)}, ..., \theta_{D(k)}\}$ are the representations of θ_D, then:

$$P(q_i|\theta_D) \quad = \quad \sum_{j=1}^{k} \lambda_j P(q_i|\theta_{D(j)})$$

We also put constraint on λ_j such that $\sum_{j=1}^{k} \lambda_j = 1$ and $\lambda_j \geq 0$ for $1 \leq j \leq k$.

In an author-oriented model, θ_D consists of three representations:

1. The document body, denoted by D
2. The author of the document D, which is represented by A_D
3. The background model from the normal case of Dirichlet prior, denoted by C.

Note that one can include the author representation be part of D. In the traditional case with Dirichlet smoothing, we use the can express $P(q_i|\theta_D)$ as:

$$P(q_i|\theta_D) \quad = \quad \lambda_1 P(q_i|D, A_D) + \lambda_2 P(q_i|C)$$

Where $\lambda_1 = \frac{|D|}{|D|+\mu}$ and $\lambda_2 = \frac{\mu}{|D|+\mu}$. If we add our author model, we would have:

$$P(q_i|\theta_D) \quad = \quad \alpha\lambda_1 P(q_i|D) + (1-\alpha)\lambda_1 P(q_i|A_D) + \lambda_2 P(q_i|C)$$

One should notice that $0 \leq \alpha \leq 1$ is a parameter to weight between the document representation and the author representation. Eventually, it would be ideal to estimate the α and λ_j automatically using the two-stage model proposed by Zhai and Lafferty[10].

Secondly, a better test set should be constructed in order to show different level of "insightfulness." Essentially, a CEES user would only need to look the top few posts in order to find the "insightful" entry that satisfy the information need. This is not shown in our setup, since we only tag whether a message is relevant to the query. Ideally, we should also tag a message as "highly"

relevant, which provides the user with the exact answer, or just "relevant", which the message only leads to a conversation that the user may be interested in.

Furthermore, more interesting work can be done in the user interface area. Showing the related messages by a highly ranked author may prove to be useful to the user of the system. It would also be desirable to cluster similar conversations automatically to generate a collection of frequently posted messages. The clusters would then give new users an quick summary of the mailing list. The users can then would have an overview on what questions have been asked before.

References

[1] Peter Eklund and Richard Cole. Structured Ontology and Information Retrieval for Email Search and Discovery. 2002.

[2] Djoerd Hiemstra. Term-Specific Smoothing for the Language Modeling Approach to Information Retrieval: The Importance of a Query Term. 2002.

[3] Rong Jin, Alex G. Hauptmann, and ChengXiang Zhai. Title Language Model for Information Retrieval. 2002.

[4] John Lafferty and ChengXiang Zhai. Document Language Models, Query Models, and Risk Minization for Information Retrieval. 2001.

[5] Paul Ogilvie and Jamie Callan. Combining Document Representations for Known-Item Search. 2003.

[6] Marc A Smith and Andrew T Fiore. Visualization Components for Persistent Conversations.

[7] Fei Song and W. Bruce Croft. A General Language Model for Information Retrieval. 1999.

[8] Chengxiang Zhai and John Lafferty. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In *Research and Development in Information Retrieval*, pages 334–342, 2001.

[9] ChengXiang Zhai and John Lafferty. Model-based Feedback in the Language Modeling Approach to Information Retrieval. 2001.

[10] ChengXiang Zhai and John Lafferty. Two-Stage Language Models for Information Retrieval. 2002.