Online Forum Thread Retrieval using Pseudo Cluster Selection and Voting Techniques

Ameer Tawfik Albaham and Naomie Salim
Faculty of Computer Science and Information System,
Universiti Teknologi Malaysia, Skudai, Johor, Malaysia
ameer.tawfik@gmail.com, naomie@utm.my

Abstract. Online forums facilitate knowledge seeking and sharing on the Web. However, the shared knowledge is not fully utilized due to information overload. Thread retrieval is one method to overcome information overload. In this paper, we propose a model that combines two existing approaches: the Pseudo Cluster Selection and the Voting Techniques. In both, a retrieval system first scores a list of messages and then ranks threads by aggregating their scored messages. They differ on what and how to aggregate. The pseudo cluster selection focuses on input, while voting techniques focus on the aggregation method. Our combined models focus on the input and the aggregation methods. The result shows that some combined models are statistically superior to baseline methods.

Keywords: Forum thread search, Voting techniques

1 Introduction

Online forums are user-generated content platforms, which enable users to build virtual communities. In these communities, users interact with each other to seek and share knowledge. The interaction happens through exchanging information in a form of discussions. A user starts a discussion by posting an initial message. Then, replies to it are contributed by the other users. A pair of an initial message and the set of its reply messages builds a thread.

Thread retrieval helps forums’ end users to find information satisfying their needs. However, the challenge is that threads are not text; they are collections of "messages"—the initial and the reply messages. Therefore, given a user query, a retrieval system needs to utilize the text of the messages to rank threads. The problem of the thread retrieval is similar to the problem of the blog site retrieval[11,6]. In blog site retrieval, given a query, we leverage the blogs’ postings in order to rank blogs. An analogy between these two retrieval problems is that threads are the blogs, and messages are the postings.

Because of the thread retrieval’s resemblance to the blog site retrieval, researchers, as in [5] and [10], have adapted techniques from blog distillation, such as [4,11], to thread search; and, the adapted models performed well. Motivated by this fact, we propose to combine two known techniques on blog site retrieval:
Pseudo Cluster Selection (PCS) \cite{11} and Voting Techniques (VT) \cite{6}; then, we apply the combined model to thread search.

Both methods first rank a list of postings and then aggregate the postings’ scores to rank their parent blogs. However, they differ in two aspects: the input and the aggregation method. PCS focuses on the top \( k \) ranked postings from each blog, whereas VT considers all ranked postings. Furthermore, PCS uses the geometric mean to fuse scores, while VT adapts various data fusion methods, such as CombMAX \cite{12} and expCombSUM \cite{9}, to the blog distillation task. In using CombMAX, blogs are ranked based on their best scoring posting; in using expCombSUM, a blog is scored based on the sum of the exponential values of the blog’s ranked postings’ relevance scores. It was reported that the voting methods that favor blogs with highly ranked postings performed better than the other voting methods \cite{6}. This characteristic is similar to PCS’s emphasis on highly ranked postings because it considers only the top \( k \) postings.

In thread retrieval, \cite{3} reported that scoring threads using the maximum score of their ranked messages is superior to using the arithmetic mean of the scores; and, PCS is statistically superior to both methods. Note that the arithmetic mean will be affected by the messages with low scores, whereas the maximum score method favors threads with highly ranked messages. In addition, CombMAX is a special case of PCS\((k = 1)\)\cite{5}. In other words, focusing on highly ranked messages improve the retrieval performance. Therefore, in this study, we hypothesize that voting methods can achieve better performance by focusing on only the top \( k \) ranked messages.

2 Related work

The voting approach to the blog distillation task is inspired by works on data fusion \cite{12,9} and expert finding \cite{7}. A data fusion technique aims to combine several ranked lists of documents generated by different retrieval methods into a unified list \cite{1}. In addition, retrieval methods that retrieved a document are voters for that document. Then, a data fusion technique is used to fuse these votes. Data fusion techniques are categorized into score based and rank based aggregation methods. The score based methods — such as CombMAX \cite{12}, use the relevance scores of documents, whereas the rank based methods, such as BordaFuse \cite{1}, utilize the ranking positions of these documents.

The expert finding task is defined as retrieving a list of people who are experts in the topic of the user query \cite{7}. To estimate the expertise of a person, methods in this task leverage the documents that are associated with or written by that person. \cite{7,7} models the problem of expert finding as a data fusion problem. Motivated by the success of the voting model in the expert finding task \cite{7,6}, models the blog site retrieval as voting process as well. The connection between these tasks is that: in both, we first rank a list of documents with respect to a user query using an underlined text retrieval model, then each ranked document is considered as a vote supporting the relevance of its associated "object"— the person or the blog. Indeed, in both tasks, the voting approach was found
Forum Thread Retrieval using Pseudo Cluster Selection & Voting Techniques

to be statistically superior to baseline methods [7,6]. Similarly, in the Pseudo Cluster Selection (PCS) method [11], we first rank a list of postings, and then an aggregation method is applied. However, in PCS, only the top $k$ ranked postings from each blog are used to estimate the blog’s relevance. If the number of the blog’s ranked postings is less than $k$, then a padding is supplied by using the minimum score of all ranked postings.

In thread retrieval, similar approach has been applied as well [5,3,10]; that is ranking threads by aggregating their message relevance scores. [5] proposed two strategies to rank threads: inclusive and selective. The inclusive strategy utilizes evidences from all messages in order to rank parent threads. Two models from previous work on blog site retrieval [4] were adapted to thread search: the large document and the small document models. The large document model creates a virtual document for each thread by concatenating the thread’s message text, then it scores threads based on their virtual document relevance to the query. In contrast, the small document model defines a thread as a collection of text units (messages). Then, it scores threads by adding up their messages’ similarity scores. In contrast to the inclusive strategy, [5]’s selective strategy treats threads as a collection of messages; and it uses only few messages to rank threads. Three selective methods were used. The first one is scoring threads using only the initial message relevance score. The second method scores threads by taking the maximum score of their message relevance scores. The third method is based on PCS. Generally, it was found that the selective models are statistically superior to the inclusive models [5,3] especially the PCS method. Our work extends the Pseudo Cluster Selection method by investigating more aggregation methods.

Another line of research is the multiple context retrieval approach [10]. This approach treats a thread as a collection of several ”local contexts” — types of self-contained text units. Four contexts were proposed: posts, pairs, dialogues and the entire thread. The thread and post contexts are identical to [5]’s virtual document and message concepts respectively. The pair and the dialogue contexts exploit the conversational relationship between messages to build text units. To rank threads using the post, pair and dialogue contexts, PCS was used. It was observed that retrieval using the dialogue context outperformed retrieval using other contexts. Additionally, the weighted product between the thread context and the post, pair or the dialogue contexts achieved better performance than using individual contexts. In our work, we are focusing on how to combine the ranked contexts’ scores — using the message context. Therefore, our work is complementary to [5]’s work.

The third line of work is the structure based document retrieval [2]. In this approach, a thread consists of a collection of structural components: the title, the initial message and the reply messages set. In this representation, the thread relevance to the user query is estimated using [8]’s inference network framework. Our work can be applied to [2]’s representation as well. We could use [2]’s inference based relevance score the same way the thread context score was used in [10].
 Combined Models

In this work, given a query \( Q = \{q_1, q_2, ..., q_n\} \), we first rank a list of messages \( R_Q \) with respect to \( Q \). Then, we score threads by aggregating their top \( k \) ranked messages' scores or ranks. [10] stated that the pair and the dialogue contexts must be extracted using thread discovery techniques, and retrieval using inaccurate extracted contexts will hurt the performance significantly. As a result, we use only the message context.

In estimating the relevance between \( Q \) and a message \( M \), we employ the query language model assuming term independence, uniform probability distribution for \( M \) and Dirichlet smoothing[13] as follows:

\[
P(Q|M) = \prod_{q \in Q} \left( \frac{n(q,M) + \mu P(q|C)}{|M| + \mu} \right)^{n(q,Q)}
\]

where \( q \) is a query term, \( \mu \) is the smoothing parameter. \( n(q,M) \) and \( n(q,Q) \) are the term frequencies of \( q \) in \( M \) and \( Q \) respectively, \( |M| \) is the number of tokens in \( M \) and \( P(q|C) \) is the collection language model.

To rank threads, the twelve aggregation methods proposed by [7] are adapted: Votes, Reciprocal Rank(RR), BordaFuse, CombMIN, CombMAX, CombMED, CombSUM, CombANZ, CombMNZ, expCombSUM, expCombANZ and expCombMNZ. In addition to these methods, this study uses ”CombGNZ”— the geometric mean of the relevance scores. We use this method because it is the aggregation method employed by the Pseudo Cluster Selection method [11].

Note that our combined models aggregate only the top \( k \) ranked messages from each thread in order to infer a thread’s relevance score. Let \( R_T \) denote the set of all ranked messages from a thread \( T \), and \( R_{T,k} \) denote the set of the top \( k \) ranked messages. However, if the size of \( R_T \) is less than \( k \), then the ”set with less members” is used— either \( R_T \) or \( R_{T,k} \), we denote to it by \( R_{T,L} \). Using \( R_{T,L} \), we can score threads using any of the following methods:

\[
Votes_k(Q,T) = |R_{T,L}|
\]

\[
RR_k(Q,T) = \sum_{M \in R_{T,L}} \frac{1}{\text{rank}(Q,M)}
\]

\[
BordaFuse_k(Q,T) = \sum_{M \in R_{T,L}} |R_Q| - \text{rank}(Q,M)
\]

\[
\text{CombMIN}_k(Q,T) = \text{MIN}_{M \in R_{T,L}} P(Q|M)
\]

\[
\text{CombMED}_k(Q,T) = \text{Median}_{M \in R_{T,L}} P(Q|M)
\]

\[
\text{CombSUM}_k(Q,T) = \sum_{M \in R_{T,L}} P(Q|M)
\]
CombANZ\_k(Q,T) = \frac{1}{|R_{T,L}|} \times \sum_{M \in R_{T,L}} P(Q|M) 

CombGNZ\_k(Q,T) = \left( \prod_{M \in R_{T,L}} P(Q|M) \right)^{\frac{1}{|R_{T,L}|}} 

CombMNZ\_k(Q,T) = |R_{T,L}| \times \sum_{M \in R_{T,L}} P(Q|M) 

expCombSUM\_k(Q,T) = \sum_{M \in R_{T,L}} \exp(P(Q|M)) 

expCombANZ\_k(Q,T) = \frac{1}{|R_{T,L}|} \times \sum_{M \in R_{T,L}} \exp(P(Q|M)) 

expCombMNZ\_k(Q,T) = |R_{T,L}| \times \sum_{M \in R_{T,L}} \exp(P(Q|M)) 

where rank(Q,M) is the rank of the message M on R\_Q, |R\_Q| is the size of R\_Q and |R\_{T,L}| is the size of R\_{T,L}. This rest of this paper reports the retrieval performance using these methods.

4 Experimental Design

Thread retrieval is a new task, and the number of test collections is limited. In this study, we used the same corpus used by [2]. It has two datasets from two forums—Ubuntu\(^1\) and Travel\(^2\) forums. The statistics of the corpus is as follows. In the Ubuntu dataset, there are 113277 threads, 676777 messages, 25 queries and 4512 judged threads. In the Travel dataset, there are 83072 threads, 590021 messages, 25 queries and 4478 judged threads. The same relevance protocol was followed: a thread with 1 or 2 relevance judgement is considered as relevant, while a relevance of 0 is considered as irrelevant. Text was stemmed with the Porter stemmer and no stopword removal was applied. In conducting the experiments, we used the Indri retrieval system\(^3\).

As for evaluation, we calculated Precision at 10 (P@10), Normalized Discounted Cumulative Gain at 10 (NDCG@10) and Mean Average Precision (MAP). In addition, we used the virtual document model(VD)\(^5\) as our baseline. We used VD because it has been used as a strong baseline in most previous studies\(^5\)\(^10\). In addition to VD, we used the basic aggregation methods. In the basic methods, all ranked messages are included in the aggregation process. That enables us to make a fair judgement about the performance of the combined methods.

\(^1\) ubuntuforum.org
\(^2\) http://www.tripadvisor.com/ShowForum-g28953-i4-New York.html
\(^3\) http://www.lemurproject.org/indri.php
As for parameter estimation, we have three parameters to estimate: the smoothing parameters $\mu$ for the virtual document and message language models, the size of the initial ranked list of messages $R_Q$ and the value of $k$. To estimate $\mu$, we varied its value from 500 up to 4000; adding 500 in each run. To estimate the size of $R_Q$, we varied its value from 500 up to 5000 adding 500 in each run. To estimate $k$—the number of top ranked messages, we varied its value from 2 up to 6 adding 1 in each run. Then, an exhaustive grid search was applied to maximize MAP using 5-fold cross validation.

**Table 1.** Retrieval performance of fusion top $k$ messages on the Ubuntu dataset.

| Method     | MAP     | P@10    | NDCG@10 |
|------------|---------|---------|---------|
| Baseline   | VD      | 0.3437  | 0.4200  | 0.3284  |
| Basic models |         |         |         |         |
| Votes      | 0.2749  | 0.4680  | 0.3551  |
| RR         | 0.3313  | 0.4600  | 0.3428  |
| BordaFuse  | 0.3153  | 0.5080  | 0.3778  |
| CombMIN    | 0.1779  | 0.2600  | 0.1840  |
| CombMAX    | 0.3074  | 0.4480  | 0.3257  |
| CombMED    | 0.2212  | 0.2760  | 0.1927  |
| CombSUM    | 0.3100  | 0.4720  | 0.3633  |
| CombMNZ    | 0.3108  | 0.4880  | 0.3720  |
| CombANZ    | 0.2314  | 0.2800  | 0.1991  |
| CombGNZ    | 0.2272  | 0.2760  | 0.1971  |
| expCombSUM | 0.3088  | 0.4840  | 0.3676  |
| expCombANZ | 0.2315  | 0.2800  | 0.1991  |
| expCombMNZ | 0.3088  | 0.4840  | 0.3676  |

| Top $k$ models |         |         |         |
|----------------|---------|---------|---------|
| Votes$_k$      | 0.2699  | 0.4600  | 0.3327  |
| RR$_k$         | 0.3359  | 0.4600  | 0.3425  |
| BordaFuse$_k$ | 0.3355  | 0.4800  | 0.3659  |
| CombMIN$_k$    | 0.2372  | 0.3320  | 0.2559  |
| CombMED$_k$    | 0.2644  | 0.3680  | 0.2698  |
| CombSUM$_k$    | 0.3575  | 0.4760  | 0.3851  |
| CombMNZ$_k$    | 0.3544  | 0.4800  | 0.3866  |
| CombANZ$_k$    | 0.2644  | 0.3680  | 0.2698  |
| CombGNZ$_k$    | 0.2668  | 0.3840  | 0.2832  |
| expCombSUM$_k$ | 0.3539  | 0.4800  | 0.3865  |
| expCombANZ$_k$ | 0.2644  | 0.3680  | 0.2698  |
| expCombMNZ$_k$ | 0.3539  | 0.4800  | 0.3865  |

5 Result and Discussion

This section reports the result of fusing only the top $k$ messages. Table 1 and Table 2 presents the retrieval performance of these methods in the Ubuntu dataset.
and the Travel dataset respectively. The symbols $\triangle$ and $\nabla$ denote statistically significant improvements or degradations over the virtual document model (VD) respectively, whereas, $\blacktriangle$ and $\blacktriangledown$ denote statistically significant improvements or degradations of a top $k$ model over its basic model, e.g CombSUM$_k$ over CombSUM. All significance tests are conducted using ttest at $p < 0.05$. The upper parts from the tables contain retrieval performance of the virtual document (VD) and the basic aggregation methods, while the lower parts contain the performance of the combined methods.

Generally, the data on these tables supports the result from previous researches. In the basic mode, BordaFuse, CombSUM, CombMNZ, expCombSUM, expCombSUM and expCombMNZ were able to produce better or comparable result with respect to VD. These methods favor threads with highly ranked messages. In contrast, CombGNZ, CombMED, CombANZ, CombMIN and exp-CombANZ might be affected by threads that have a lot of low scored messages. In the combined models, all rank based methods have almost similar results to their results on the basic mode. In contrast, the score based methods benefit largely from fusing only the top $k$ messages. Among them, methods favouring threads with highly ranked messages brought significant improvements over baseline methods— see the performance of CombSUM, CombMNZ, expCombSUM and expCombMNZ. In addition, CombANZ, CombMIN, CombMED and expCombANZ benefit as well. Therefore, focusing on top $k$ messages is a good strategy to improve data fusion performance on thread retrieval.

Discussion

The improvement might be due to the removal of noise introduced when all ranked messages are considered. By focusing on the top $k$ messages, low score messages are discarded leading to improving retrieval. In fact, ranking using only one message, e.g. CombMAX, might not be enough to capture the topical relevance of threads, while considering all ranked messages might introduce irrelevant messages. Therefore, focusing on the top $k$ ranked messages will balance these two aspects. This premise explains, to some extend, why no improvements were observed among rank based methods. Rank based methods inherently address the issue of low scoring messages. Furthermore, the near similar performance of CombSUM, CombMNZ and their exponential variants suggests that: once we focus on the top $k$ messages, extra emphasis on these messages does not help. In other words, focusing on the top ranked messages is the main reason behind the observed improvements.

To confirm this hypothesis, a study was conducted to investigate how does the performance change as $k$ increases? As Figures 1 and 2 show, the optimal value of $k$ for each method ranges between 2 to 5 messages.Going beyond the optimal value, rank based methods tend to have consistent performances that are similar to their result on the basic mode. In contrast, score based methods performed badly as $k$ increases. That further supports the proposed hypothesis. Another interesting observation is the performance of CombGNZ. The result
Table 2. Retrieval performance of fusion top \( k \) messages on the Travel dataset.

| Method            | MAP     | P@10    | NDCG@10 |
|-------------------|---------|---------|---------|
| Baseline          | VD      |         |         |
| Votes             | 0.2996  | 0.5280  | 0.4190  |
| RR                | 0.3155  | 0.4520  | 0.3432  |
| BordaFuse         | 0.3630  | 0.5640  | 0.4350  |
| CombMIN           | 0.1574  | 0.3040  | 0.2199  |
| CombMAX           | 0.2727  | 0.4360  | 0.3216  |
| Basic models      |         |         |         |
| CombMED           | 0.2004  | 0.3480  | 0.2380  |
| CombSUM           | 0.3668  | 0.5560  | 0.4440  |
| CombMNZ           | 0.3575  | 0.5280  | 0.4205  |
| CombANZ           | 0.2065  | 0.3400  | 0.2346  |
| CombGNZ           | 0.2000  | 0.3320  | 0.2319  |
| expCombSUM        | 0.3513  | 0.5200  | 0.4109  |
| expCombANZ        | 0.2065  | 0.3400  | 0.2346  |
| expCombMNZ        | 0.3513  | 0.5200  | 0.4109  |
| Top \( k \) models |         |         |         |
| Votes\(_k\)      | 0.2864  | ▼ 0.5280 | 0.3943  |
| RR\(_k\)         | 0.3126  | ▼ 0.4520 | 0.3400  |
| BordaFuse\(_k\)  | 0.3639  | ▼ 0.5520 | 0.4380  |
| CombMIN\(_k\)    | 0.1918  | ▲ 0.3360 | ▲ 0.2381 |
| CombMED\(_k\)    | 0.2478  | ▲ 0.3960 | ▲ 0.2881 |
| CombSUM\(_k\)    | 0.3766  | ▲ 0.6080 | ▲ 0.4788 |
| CombMNZ\(_k\)    | 0.3608  | ▲ 0.5720 | ▲ 0.4587 |
| CombANZ\(_k\)    | 0.2478  | ▲ 0.3960 | ▲ 0.2882 |
| CombGNZ\(_k\)    | 0.2330  | ▲ 0.3760 | ▲ 0.2754 |
| expCombSUM\(_k\) | 0.3600  | ▲ 0.5720 | ▲ 0.4550 |
| expCombANZ\(_k\) | 0.2479  | ▲ 0.3960 | ▲ 0.2882 |
| expCombMNZ\(_k\) | 0.3600  | ▲ 0.5720 | ▲ 0.4550 |

Fig. 1. The performance of the combined methods as \( k \) changes on the Ubuntu dataset.
shows the inferiority of CombGNZ to the aforementioned good performing methods. However, [5,3] reported that PCS—using the geometric mean, outperformed CombMAX. That does not contradict our findings. In [5,3], PCS adds a padding step if the number of ranked messages is less than $k$, whereas we do not apply this step. We plan to study the effect of the padding step in future works.

6 Conclusion and Future Works

In this paper, we addressed the problem of online forums thread retrieval. We conducted experiments to investigate the performance of combining the pseudo cluster selection and voting techniques approaches. The results showed that the combined models can improve the performance of score based voting techniques significantly in various measures.

Our future work has two directions. First, we will apply the combined methods as aggregation methods on the other thread representations such as fusing the dialogue or the pair relevance scores [10]. Second, we will study the effect of the padding step on the performance of the combined models.

Acknowledgements This work is supported by Ministry of Higher Education (MOHE) and Research Management Centre (RMC) at the Universiti Teknologi Malaysia (UTM) under Research University Grant Category ( VOT Q. J130000 .2528 .02H99 ).

References

1. J. A. Aslam and M. Montague. Models for metasearch. In Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR ’01, pages 276–284, New York, NY, USA, 2001. ACM.
2. S. Bhatia and P. Mitra. Adopting inference networks for online thread retrieval. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, pages 1300–1305, Atlanta, Georgia, USA, July 11–15 2010.

3. J. L. Elsas. Ancestry.com online forum test collection. Technical Report CMU-LTI-017, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 2011.

4. J. L. Elsas, J. Arguello, J. Callan, and J. G. Carbonell. Retrieval and feedback models for blog feed search. In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, SIGIR ’08, pages 347–354, New York, NY, USA, 2008. ACM.

5. J. L. Elsas and J. G. Carbonell. It pays to be picky: an evaluation of thread retrieval in online forums. In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, SIGIR ’09, pages 714–715, New York, NY, USA, 2009. ACM.

6. C. Macdonald and I. Ounis. Key blog distillation: ranking aggregates. In Proceedings of the 17th ACM conference on Information and knowledge management, CIKM ’08, pages 1043–1052, New York, NY, USA, 2008. ACM.

7. C. Macdonald and I. Ounis. Voting techniques for expert search. Knowl. Inf. Syst., 16(3):259–280, Aug. 2008.

8. D. Metzler and W. B. Croft. Combining the language model and inference network approaches to retrieval. Inf. Process. Manage., 40(5):735–750, Sept. 2004.

9. P. Ogilvie and 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, SIGIR ’03, pages 143–150, New York, NY, USA, 2003. ACM.

10. J. Seo, W. Bruce Croft, and D. Smith. Online community search using conversational structures. Information Retrieval, 14:547–571, 2011. 10.1007/s10791-011-9166-8.

11. J. Seo and W. B. Croft. Blog site search using resource selection. In Proceedings of the 17th ACM conference on Information and knowledge management, CIKM ’08, pages 1053–1062, New York, NY, USA, 2008. ACM.

12. J. A. Shaw, E. A. Fox, J. A. Shaw, and E. A. Fox. Combination of multiple searches. In The Second Text REtrieval Conference (TREC-2), pages 243–252, 1994.

13. C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. ACM Trans. Inf. Syst., 22(2):179–214, Apr. 2004.