Retrieval and Feedback Models for Blog Distillation

Jonathan Elsas, Jaime Arguello, Jamie Callan, Jaime Carbonell
Language Technologies Institute, School of Computer Science, Carnegie Mellon University
Pittsburgh, PA, 15213
{jelsas,jaime,callan,jgc}@cs.cmu.edu

Abstract

This paper presents our system and results for the Feed Distillation task in the Blog track at TREC 2007. Our experiments focus on two dimensions of the task: (1) a large-document model (feed retrieval) vs. a small-document model (entry or post retrieval) and (2) a novel query expansion method using the link structure and link text found within Wikipedia.

1 Introduction

Blog distillation (or "feed search") is the task of finding blog feeds with a principle, recurring interest in X, where X is some information need expressed as a query. Thus, the input to the system is a query and the output is ranked list of blog feeds. Tailoring a system for feed search requires making several design decisions. In this work, we explored two different decisions:

1. Is it most effective to treat this task as feed retrieval, viewing each feed as a single document; or entry retrieval, where ranked entries are aggregated into an overall feed ranking?

2. How can query expansion be appropriately performed for this task? Two different approaches are compared. The first one is based on pseudo-relevance feedback using the target collection. The second is a simple novel technique that expands the query with ngrams obtained from Wikipedia hyperlinks. Exploiting corpora other than the target corpus for query expansion has proven a valuable technique, especially for expanding difficult queries [7].

The four runs submitted to the Blog Distillation task correspond to varying both of these dimensions. Throughout our experiments, all retrieval was done with the Indri[1] retrieval engine using only terms from the topic *title*.

The remaining of this paper is organized as follows. Section 1 describes the pre-processing steps. Section 2 describes the target-corpus- and Wikipedia-based query expansion techniques. Section 3 describes the two retrieval models used, as well as our methods of parameter selection for the different features used in those models. Experimental results and analysis are presented in Section 4.

2 Corpus Pre-processing

For all of the runs submitted, we only used the information contained within the `feed` documents. The BLOG06 collection contains approximately 100k feed documents, which are a mix of ATOM and RSS XML. These two formats contain different XML elements which were mapped to a unified representation in order to make use of the structural elements within the feeds. We used the Universal Feed Parser[2] pack-

[1] http://www.lemurproject.org
[2] http://feedparser.org/

age for Python[3] to abstract the different data elements across all feed types to a single universal representation. For details on the mapping between ATOM and RSS elements refer to the Universal Feed Parser documentation. Documents were stemmed using the Krovetz stemmer and common stop words were removed as well as manually identified web- and feed-specific stop words such as "www", "html" and "wordpress". We filtered documents that were self-identified as non-English (in their `feed.lang` or `channel.language` elements) and feeds with fewer than 4 posts.

3 Two Query Expansions Models

Query expansion is a well-studied technique used in ad hoc retrieval to improve retrieval performance, particularly for queries with insufficient content. On the TREC 2007 Blog Distillation task, the average number of words per topic title[4] was 1.91. Expanding such terse queries with as many relevant terms has a strong potential for improving precision and recall.

3.1 Indri's relevance model

Our first query expansion feature used Indri's built-in facilities for pseudo-relevance feedback [2, 3, 4]. To generate our query expansion terms, we constructed a Full Dependence Model query [5, 4] with the terms in the topic *title*.[5] For all of our submissions, this query was run against the entire indexed feeds and did not take advantage of any indexed document structure. In preliminary experimentation this yielded the best results. Using this query, $N = 10$ documents were retrieved and a relevance model was built with those returned results. The top $k = 50$ most likely terms were extracted from that relevance model, and these terms constituted our relevance model query Q_{RM}. This query was then used as a feature for our unified feed and entry queries. N and k were set to values that had previously been shown to be effective for pseudo-relevance feedback in other tasks [4].

[3]http://www.python.org/

[4]Only text from the topic *title* was used to query the system on all 4 runs submitted to the track, as the topic title more closely resembles the types of queries a real user might submit to a search engine compared to compared to the topic *description* and *narrative*

[5]Throughout these experiments, parameters for the full dependence model queries were set identically to [5]: 0.8 for the unigram feature weights and 0.1 for the window and proximity feature weights.

3.2 Wikipedia for query expansion

Some prior work has explored using using Wikipedia for query expansion. In [1], Collins-Thompson and Callan combine term association evidence from WordNet[6] and Wikipedia[7] in a Markov chain framework for query expansion. In [8], Li et al. use Wikipedia for query expansion more directly. In their algorithm, as in our approach, each test query was run on both the target corpus and Wikipedia. Wikipedia articles were ranked differently, however, utilizing article metadata unique to Wikipedia. Each Wikipedia article belongs to one or more categories. A weight W_c was assigned to each category c based on the number of articles belonging to c ranking among the top 100. Then, each Wikipedia article d was ranked by a linear combination of its original document score and $W_d = \sum_{cat(d) \in c} W_c$, the sum of the weights W_c for each category c to which d belongs. Twenty expansion terms were selected ad hoc from the top 40 Wikipedia articles.

Wikipedia articles are available for download in their original markup language, called *Wikitext*, which encodes useful metadata such as the article's title and its hyperlinks. Each hyperlink contains both the title of the target page and optional anchor text. In cases where no anchor text is specified, it resolves to the title of the target page. During preprocessing, a sample of about 650, 000 articles from the English portion of the Wikipedia were indexed using Indri. Our simple

[6]http://wordnet.princeton.edu/

[7]http://www.wikipedia.org/

algorithm was motivated by the observation that valuable expansion ngrams are contained in hyperlinks pointing to Wikipedia articles that are relevant to the base query.

First, the seed query was run against the Wikipedia corpus. The top N ranked articles were added to set D_N. Then, all anchor phrases used in hyperlinks in D_N were added to set $A_{(D_N)}$. Note that an anchor phrase a_i in $A_{(D_N)}$ may occur several times in D_N and different occurrences of a_i need not be associated with hyperlinks to the same article. For example, suppose that the seed query is *space exploration*. Within the top N articles, the phrase *NASA* may occur several times as the anchor text of a hyperlink. Some of these hyperlinks may link to the Wikipedia article on the *National Aeronautics and Space Administration*, while others may link to the Wikipedia article on the *NASA Ames Research Center*, and so forth. A single occurrence of anchor phrase a_i is denoted as a_{i_j} and the target article of the hyperlink anchored by a_{i_j} is denoted as $target(a_{i_j})$. Each anchor phrase a_i was scored according to

$$score(a_i) = \sum_{a_{i_j} \in A_{(D_N)}} \Big[\mathbb{I}(\text{rank}(\text{target}(a_{i_j})) \leq T) \times (T - \text{rank}(\text{target}(a_{i_j})))\Big].$$

The identity function $I(\cdot)$ equals 1 if $rank(target(a_{i_j})) \leq T$ and 0 otherwise. Intuitively, the score of anchor ngram a_i is highest when the hyperlinks anchored by a_i link to many articles that are ranked highly against the seed query. In our runs, $N = 1000$ and $T = 500$, and were selected ad hoc. Anchor ngrams occurring less than 3 times were ignored and the 20 top scoring anchor ngrams were selected. Their scores were normalized to sum to 1 and each ngram's normalized score was used to weight it with respect to the other expansion ngrams.

N and T may seem large compared to parameters typical of PRF. Intuitively, N and T play different roles. N controls the size of the search space. T controls the range of topical aspect of the ngrams considered. Thus, a large N and a small T increases the chance of finding synonyms or paraphrases of the same concept by focusing on many anchor ngrams that link to the same highly-ranked article. With larger values of T, it is expected that ngrams will relate to a wider range of topics. Larger values of T also increase the risk of extracting irrelevant ngrams. By setting N and T large in our runs, we aim for high synonym variability and broad topical aspect coverage. One natural question is how sensitive this method is to parameters N and T. Ultimately, anchor ngrams are scored proportional to the rank of their hyperlink's target page. Initial experiments varying N and T with T sufficiently large (≥ 100) showed stability in the top ranked expansion phrases.

Finally, the resulting query, Q_W is given by:

$$\texttt{\#weight(} \quad score(a_1)\, DM_{a_1} \quad score(a_2)\, DM_{a_2} \quad \ldots \quad score(a_{20})\, DM_{a_{20}})$$

where DM_{a_i} is the Full-dependence model query formed with the anchor text ngram, a_i.

4 Two Retrieval Models

As described above, we investigated two models of feed retrieval: (1) the *large document* approach, where each feed was treated as a single document and then ranked in the typical fashion, and (2) the *small document* approach, where posts were the unit of retrieval and feeds were ranked based on the quantity and quality of their retrieved posts. The following sections describe these two approaches in detail and our method for selecting parameters used in these models.

4.1 Large Document Model

The large document approach does not distinguish between post content and number of posts within a feed. We did retain the structural elements present in feeds such as `feed.title` and `feed.entry`, but treated these as features of the monolithic feed document. We performed retrieval in a standard fashion on these feed documents, utilizing the feed and entry structural elements.

The query features used in the large document model are given below:

- Full Dependence Model on the `feed.title` field (DM_T),
- Full Dependence Model on the `feed.entry` field(s) (DM_E),
- Indri's Relevance Modeling PRF (Q_{RM}), and
- Wikipedia-based expansion (Q_W).

and the final Indri query is as follows:

$$\#\texttt{weight}(\quad \lambda_T\ DM_T \quad \lambda_E\ DM_E \\ \lambda_{RM}\ Q_{RM} \quad \lambda_W\ Q_W)$$

where $\lambda_i > 0, \sum \lambda_i = 1$.

4.2 Small Document Model

The small document model views the feeds as different document collections and the entries as documents within those collections. Under this framework, the feed retrieval task can be seen as analogous to that of resource selection or ranking in federated search – given a query, find the document collections most likely to contain relevant documents.

Our approach to resource ranking was similar to the Relevant Document Distribution Estimation (ReDDE) [6]. In that approach, given known (true or estimated) collection sizes and a database of sampled documents from all collections, collections are ranked by retrieving from the sampled database and summing the document scores from that sampled retrieval. The basic ReDDE resource scoring formula for collection C_j is:

$$\hat{Rel}_q(j) = \sum_{d_i \in C_j} P(rel|d_i) P(d_i|C_j) N_{C_j}$$

where $P(rel|d_i)$ is the probability of document relevance for the query, $P(d_i|C_j)$ is the probability of selecting the document from collection C_j (or, as is typically the case, from our sampled version of the true collection), and N_{C_j} is the size of the collection.

To support a simplified federated search model of feed retrieval, we chose to create a new collection by sampling the posts from each feed. The BLOG06 corpus contains feeds ranking in size from just 1 or 2 posts to feeds with several hundred. Figure 1 illustrates the distribution of feed sizes in the corpus.

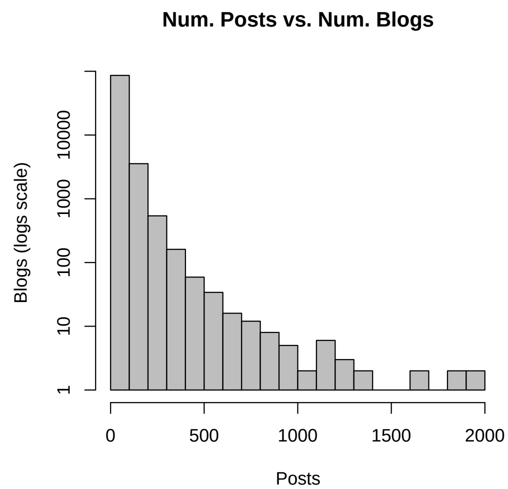

Figure 1: Blog size distribution

When creating the corpus for our federated search model, we sampled 100 posts per feed (with replacement), letting us assume a uniform $N_{C_j} = 100\ \forall j$. Assuming all posts are equally likely to be retrieved for each feed, i.e. $P(d_i|C_j) = 1/100$, the above resource scoring formula simplifies to:

$$\hat{Rel}_q(j) = \sum_{d_i \in C_j} P(rel|d_i)$$

There is one difference between our approach and the ReDDE approach. In the ReDDE approach, N_{C_j} is the size of the original collection (possibly estimated), and not the size of the sampled collection N'_{C_j}. Here, we set $N_{C_j} = N'_{C_j} = 100\ \forall j$.

By doing so, a feed's original, pre-sampled size does not directly factor into the scoring function. This choice was made because the goal of the task is to find feeds with a central interest in some topic X, irrespective of feed size.

The scoring function, $\hat{Rel}_q(j)$, can be easily expressed in the Indri query language:

$$\#\texttt{wsum}(\ 1.0\ \#\texttt{combine}[\texttt{entry}](\ Q_E\ ))$$

where Q_E is our entry query, the inner `#combine[entry]` produces scores over entries within a single feed, and the outer `#wsum` adds these scores to generate a feed-level score. Note that there is no `#sum` operator in the Indri query language, necessitating the constant 1.0 in the query, which doesn't have any effect on the final ranking.

The entry query Q_E used the same pseudo-relevance feedback features described above:

$$\#\texttt{weight}(\lambda_E\ DM_E\ \ \lambda_{RM}\ Q_{RM}\ \ \lambda_W\ Q_W)$$

4.3 Parameter Selection

The above queries have a number of free parameters that must be chosen appropriately for effective retrieval. To do this, we selected a small subset of the queries (956, 964, 966, 986, 989, 991 and two others not included in the evaluation), performed initial retrieval experiments using simple bag-of-words queries, and judged the top 50 documents retrieved as relevant/non-relevant. We used this small training set to tune our parameters via a simple grid-search. Table 1 gives the parameter settings that maximized mean average precision for all runs using both retrieval models and different pseudo relevance feedback features.

Model	PRF	λ_T	λ_E	λ_{RM}	λ_W
large-doc	RM	0.2	0.6	0.2	–
	RM+W	0.2	0.3	0.1	0.4
small-doc	RM	–	0.3	0.7	–
	RM+W	–	0.3	0.5	0.2

Table 1: Query weight settings. RM=Relevance Model PRF, W=Wikipedia PRF

Although we used a small subset of the evaluation queries to train our system, we do not believe our results are strongly biased towards these queries. Our best run's performance (CMUfeedW) on these training queries was highly variable, achieving the best performance on only one of the training queries (989) and close to our worst performance on several others. Figure 2 shows the performance of this run with the training queries clearly indicated.

4.4 Results & Discussion

Table 2 shows the performance of our four runs: large document (CMUfeed) vs. small document (CMUentry) retrieval models and the Wikipedia (*W) expansion model. The large document model clearly outperformed the small document model, and Wikipedia-based expansion improved average performance of all runs. Figure 2 shows our best run (CMUfeedW) compared to the per-query best and median average precision values.

Run	MAP	R-prec	P10
CMUfeed	0.3385	0.4087	0.4733
CMUfeedW	**0.3695**	**0.4245**	**0.5356**
CMUentry	0.2453	0.3277	0.4089
CMUentryW	0.2552	0.3384	0.4267

Table 2: Performance of our 4 runs

Retrieval performance was superior with Wikipedia-based query expansion than without. Adding Wikipedia-based expansion improved performance in 30/45 queries under the *small document* model and 34/45 queries under the *large document* model. The largest improvement under both document models, based on average precision, was for query 967, *home baking*. An improvement of 6,780% was achieved under the *small document* model and 683% under the *large document* model. Table 3, shows the expansion terms obtained in descending order of confidence for both retrieval models.

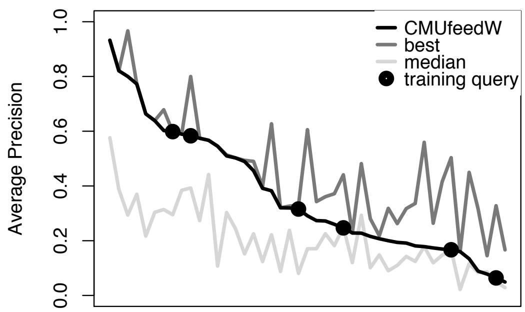

Figure 2: Best & Median AP per query compared to CMUfeedW (ordered by CMUfeedW).

Wikipedia	**Relevance Model**
bread	home
baking	business
flour	base
butter	2005
yeast	work
cake	start
baking powder	job
cookie	shoe
carbon dioxide	portal
honey	bread

Table 3: Top 10 expansion terms/phrases for topic 967, *home baking*, for both of our expansion models.

One limitation of our Wikipedia-based approach is that its parameters (e.g., the number of expansion terms) remain constant irrespective of the seed query. This is troublesome in cases where the topic drifts rapidly down the ranked list of Wikipedia articles.

In conclusion, our experimental results showed that the *large document* approach outperformed the *small document* approach for this task. Additionally, the simple method of finding query expansion terms and phrases from Wikipedia proved to be effective across runs. The two retrieval models and the Wikipedia feedback model present interesting research questions. Alternate sampling and rank aggregation methods may improve the performance of the *small document* model. The use of anchor text for query expansion could be explored further, beyond Wikipedia and feed distillation.

Acknowledgments

This work was supported in part by the eRulemaking project and NSF grant IIS-0240334, and DARPA contract IBM W0550432 (DARPA PRIME Agreement # HR0011-06-2-0001).

References

[1] Kevyn Collins-Thompson and Jamie Callan. Query expansion using random walk models. In *Proc of CIKM '05*, 2005.

[2] Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *Proc of SIGIR 01*, pages 120–127, New York, NY, USA, 2001. ACM Press.

[3] D. Metzler, T. Strohman, H. Turtle, and W. Croft. Indri at trec 2004: Terabyte track. In *Proc of TREC 04*, 2004.

[4] D. Metzler, T. Strohman, Y. Zhou, and W. Croft. Indri at trec 2005: Terabyte track. In *Proc of TREC 05*, 2005.

[5] Donald Metzler and W. Bruce Croft. A markov random field model for term dependencies. In *Proc of SIGIR 05*, pages 472–479, New York, NY, USA, 2005. ACM Press.

[6] Luo Si and Jamie Callan. Relevant document distribution estimation method for resource selection. In *Proc of SIGIR 03*, pages 298–305, New York, NY, USA, 2003. ACM Press.

[7] Ellen M. Voorhees. Trec report: The trec robust retrieval track. In *Proc of ACM SIGIR Forum*, 2005.

[8] E.K.S Ho Y. Li, R.W.P. Luk and F.L. Chung. Improving weak ad-hoc queries using wikipedia as external corpus. In *Proc of SIGIR '07*, 2007.