University of Lugano at TREC 2010

Mostafa Keikha, Parvaz Mahdabi , Shima Gerani, Giacomo Inches , Javier Parapar†,
Mark Carman and Fabio Crestani
University of Lugano
Department of Informatics
Lugano, Switzerland
{mostafa.keikha, parvaz.mahdabi, shima.gerani, giacomo.inches, mark.carman,
fabio.crestani}@usi.ch
†Information Retrieval Lab
†Dept. of Computer Science
†University of A Coruna
†javierparapar@udc.es

ABSTRACT

We report on the University of Lugano's participation in the Blog and Session tracks of TREC 2010. In particular we describe our system for performing blog distillation, faceted search, top stories identification and session reranking.

1. INTRODUCTION

User generated content has recently become one of the most important sources of information in the web. This data contains a lot of information regarding users' opinions and experiences, which can be useful in many applications. Forums, on-line discussions, community question answering sites and social networks are all examples of such content. Blogs are another important source of information in this category and are the content of interest for the TREC (Text REtrieval Conference) blog track [5, 8], which involves tasks of faceted blog distillation and top story identification. We explain our approach to faceted blog distillation in section 2 and to top stories identification is explained in section 3. The methods we used for the TREC session track are described in section 4 and we provide conclusions in section 5.

2. BOG DISTILLATION

Blog distillation is the problem of retrieving blogs that are relevant to a given query. Retrieving blogs as collections of documents, as apposed to retrieving single documents in traditional IR tasks, is the main characteristic of this problem. We employ the Blogger Model for retrieving blogs for the distillation task [1]. In this model, blogs are ranked by :

$$p(blog|q) \propto p(q|blog)p(blog) \tag{1}$$

where the blog prior $p(blog)$ is estimated by a uniform distribution and the query likelihood $p(q|blog)$ is calculated as follows:

$$p(q|blog) = \prod_{t \in q} \widehat{p}(t|blog)^{n(t,q)} \tag{2}$$

$$\widehat{p}(t|blog) = \lambda p(t|blog) + (1-\lambda)p(t) \tag{3}$$

$$p(t|blog) = \sum_{post \in blog} \widehat{p}(t|post)p(post|blog) \tag{4}$$

where again $p(post|blog)$ is assumed to be uniform.

$$\widehat{p}(t|post) = \beta p(t|post) + (1-\beta)p(t) \tag{5}$$

This post level smoothing is the only difference between our implementation and the original Blogger Model. $p(t|post)$ is estimated by Maximum Likelihood estimation of the term probability the post. $p(t)$ denotes the probability of the term in the collection.

2.1 Faceted Search

For the faceted rankings, we followed our approach from last year with a small modification which we explain below. We first generated positive and negative facet scores for each retrieved blog, denoted $pos(b)$ and $neg(b)$ respectively. Facet scores for each blog are calculated using only the posts that are relevant to the query, (i.e. those in the top 15000 posts for the query). A blog facet score is calculated based on the average of the facet score for the relevant posts:

$$score_{\texttt{facet}}(b) = E_b[score_{\texttt{facet}}(d)] = \sum_{d \in b} p(d|b) score_{\texttt{facet}}(d) \tag{6}$$

We consider uniform the probability over posts for a blog, (i.e. $p(d|b) = 1/|posts|$). These facet scores induce a ranking, denoted $r_{\texttt{pos}}(b,q)$ and $r_{\texttt{neg}}(b,q)$, which we combined with the original relevance ranking $r_{\texttt{rel}}(b,q)$ using the Borda Fuse aggregation method as follows:[1]

$$score_{\texttt{BF}}(d,q) = \alpha\ r_{\texttt{rel}}(d,q) + (1-\alpha)\ r_{\texttt{facet}}(d,q) \tag{7}$$

[1]Note that whenever there are ties in the ranking, (i.e. blogs b_1 and b_2 have the same score), then the rank for those blogs is the average of the (total order) ranking.

.

	Indepth	Shallow
stdBaseline1	0.3046	0.1425
CrossEntropy	0.3056	0.1280
stdBaseline2	0.2176	0.1033
CrossEntropy	0.2176	0.0957
stdBaseline3	0.1747	0.874
CrossEntropy	0.1747	0.0856

Table 1: MAP of the In-depth vs. Shallow facet re-ranking over the three TREC baselines for all queries

In order to enable fair comparison between different facet re-ranking methods, in TREC 2010, the organizers distributed three standard baselines which are the results of the first phase (blog distillation) among participants. Participants could submit up to four runs for each of the standard baselines. We re-ranked the standard baselines using our facet scoring methods and used the TREC 2009 data (i.e. relevance judgments) for training in order to set an appropriate value for the weighting coefficient α.

2.1.1 In-depth versus Shallow

For the in-depth versus shallow facet, we calculated the Cross Entropy (CE) between each retrieved document(post) and the collection as a whole. We used CE as the positive score for the positive (*in-depth*) facet value since high CE indicates that the document contains many rare and informative words:

$$pos(d) = CE(p(.|d), p(.|c)) = \sum_{t \in d} p(t|d) \log \frac{1}{p(t|c)} \qquad (8)$$

Here $p(t|d)$ is the probability of a term t appearing within the document d, which we calculate using the relative term frequency as follows: $p(t|d) = \mathtt{tf}(t,d)/\sum_{t'} \mathtt{tf}(t',d)$, where $\mathtt{tf}(t,d)$ is the absolute term frequency. Meanwhile $p(t|c)$ denotes the probability of a term across the whole collection c, for which we use a document frequency based estimate $p(t|c) = \mathtt{df}(t)/|c|$ where $|c|$ is the number of documents in the collection. Our rational for using a `df` rather than `tf` based estimate is that the former appears less susceptible to noise from spam documents, which oftentimes include terms with very high frequency (high `tf` values).

For the negative (*shallow*) facet score we simply use the negation of the CE, i.e. $neg(d) = -pos(d)$. Table 1 shows the result of re-ranking the baselines with this method.

2.1.2 Opinion versus Factual

For the opinion versus factual facet, we built lexicons of opinionated and objective words using the TREC Blog06 collection and corresponding relevance/opionion judgments. In the lexicon, terms were weighted according to a document-frequency based version of the Mutual Information (MI) metric [6]. We then calculated (positive and negative) facet scores for each retrieved document by averaging over the lexicon weights for each word in the document (see equation 11 below.)

In order to calculate both positive (*opinionated*) and negative (*factual*) facet weights for terms we split the Mutual Information metric into two values as follows. Let $\mathcal{T}$ denote the event that a document contains the particular term t, and $\bar{\mathcal{T}}$ the event that the document doesn't contain the term. Then let $\mathcal{O}$ denote the event that a document is classed as being (relevant and) opinionated about the query and $\bar{\mathcal{O}}$ that it is (relevant but) not opinionated about the query. We calculate the positive facet score for a term by calculating the MI summation only over the two positively correlated quadrants (i.e. $\mathcal{T} \cap \mathcal{O}$ and $\bar{\mathcal{T}} \cap \bar{\mathcal{O}}$) as follows:

$$pos(t) = p(\mathcal{T}, \mathcal{O}) \log \frac{p(\mathcal{T}, \mathcal{O})}{p(\mathcal{T}), p(\mathcal{O})} + p(\bar{\mathcal{T}}, \bar{\mathcal{O}}) \log \frac{p(\bar{\mathcal{T}}, \bar{\mathcal{O}})}{p(\bar{\mathcal{T}}), p(\bar{\mathcal{O}})} \qquad (9)$$

The negative facet score is calculated analogously as follows:

$$neg(t) = p(\mathcal{T}, \bar{\mathcal{O}}) \log \frac{p(\mathcal{T}, \bar{\mathcal{O}})}{p(\mathcal{T}), p(\bar{\mathcal{O}})} + p(\bar{\mathcal{T}}, \mathcal{O}) \log \frac{p(\bar{\mathcal{T}}, \mathcal{O})}{p(\bar{\mathcal{T}}), p(\mathcal{O})} \qquad (10)$$

We calculate the required joint and marginal probabilities using document frequency estimates using the sets of *opinionated* O and *relevant* R documents in the TREC Blog06 collection as:

$$p(\mathcal{T}, \mathcal{O}) = \mathtt{df}(t, O)/|R|$$
$$p(\mathcal{T}) = \mathtt{df}(t, R)/|R|$$
$$p(\mathcal{O}) = |O|/|R|$$

Where $\mathtt{df}(t, O)$ is the number of opinionated documents containing the term t. The other joint and marginal probabilities required for equations 9 and 10 are estimated analogously.

Having calculated positive and negative weights for each term, we then averaged these lexicon weights over each document to calculate positive and negative facet scores for the document as follows:

$$pos(d) = E_d[pos(t)] = \sum_{t \in d} p(t|d) pos(t) \qquad (11)$$

As an alternative to the lexicon built from the Blog06 collection, for another set of runs, we used a lexicon which is built from amazon review and specification corpus [3]. We considered p(subj|t) provided by the lexicon as pos(t) in equation 11.

We should note that we also investigated using the number of comments and emoticons in each post as two additional features (besides the opinion score) and trained a SVM classifier using last year's data, but they didn't show to be useful in training a good classifier.

Table 2 shows the result of re-ranking the baselines with the MI weight of terms in the Blog06 collection and the second method in which we used an opinion lexicon.

2.1.3 Personal versus Official

Finally for the personal versus official facet, the same scores were used as in the opinion case, since we believe that more "personal content" is on the whole more likely to contain opinions than more "official content".

Table 3 shows the result of re-ranking the baselines with the MI weight of terms in the Blog06 collection and the second method in which we used an opinion lexicon.

3. TOP STORIES IDENTIFICATION

Our method for the top stories task proceeded as follows. We first extracted time-stamped blog posts for each query date. Applying a clustering method on the extracted posts

	Opinionated	Factual
stdBaseline1	0.1888	0.2189
MI	0.1926	0.2155
Lexicon	0.1974	0.1888
stdBaseline2	0.1274	0.1757
MI	0.0976	0.1774
Lexicon	0.1415	0.1201
stdBaseline3	0.1085	0.1058
MI	0.0987	0.1058
Lexicon	0.1225	0.1020

Table 2: MAP of the Opinionated vs. Factual facet re-ranking over the three TREC baselines for all queries

	Personal	Official
stdBaseline1	0.1973	0.2457
MI	0.2573	0.2469
Lexicon	0.1920	0.2469
stdBaseline2	0.1442	0.1832
MI	0.1529	0.1763
Lexicon	0.1429	0.1583
stdBaseline3	0.0857	0.1956
MI	0.0839	0.1956
Lexicon	0.0848	0.1956

Table 3: MAP of the Personal vs. Official facet re-ranking over the three TREC baselines for all queries

for each date, we generate different topics that have been discussed on each day. Then by considering the importance of each extracted topic in the Reuters headline corpus, we generate a ranked list of headlines for each day. In the following sections we outline our approach in more detail.

3.1 The algorithm

In this section we present details of our top stories identification algorithm. Our method for top stories task proceeds as follows.

1. For every query date we extract a set of blog posts that have a publish date that is the same as query.

2. We cluster the selected posts for each day in multiple clusters, by assuming each cluster would be about some topics that have been discussed on that date. Details of our clustering method are presented in Section 3.2.

3. Using the KL divergence between each cluster and the collection, we extract the most informative terms for each cluster.

4. The informative terms for each cluster are used as a query against the headlines collection in order to extract the most important headlines for each cluster.

5. Aggregating the ranked lists of headlines for each date, we generate our final ranking of the headlines.

6. We classify the headlines into the pre-defined classes.

3.2 Clustering the blog posts

For each query date, we selected a random sample containing 10% of the blog posts published on the date of the query and then followed a simple clustering process based on k-means [7]. We decided to use k-means clustering because of its low computational requirements and the time constraints that we had. The clustering process was run for every query date independently. For each date, we randomly selected 100 documents from among the sample of blog posts to act as seeds for the clustering algorithm. After running the k-means algorithm over the 10% sample for every query, we had a set of 100 clusters from which we extracted 100 topics as explained in the next section.

In the k-means algorithm three factors need to be determined: the document representation, the similarity measure and the stop condition:

1. Every document d, a blog post, was represented by its $tfidf$ vector, in which the value for each term t was calculated as follows:

$$tfidf(d,t) = tf(d,t) \times \log(\frac{N}{df(t)})$$

where $tf(d,t)$ is the term frequency of the term t in the document d, $df(t)$ is the document frequency of the term t in the collection sampling of size N.

2. In order to compute the documents similarities in the clustering method we used the cosine similarity:

$$Sim(d_i,d_j) = \frac{\sum_{t=1}^{m} tfidf(d_i,t) \times tfidf(d_j i,t)}{\sqrt{\sum_{t=1}^{m} tfidf(d_i,t)^2 \sum_{t=1}^{m} tfidf(d_j,t)^2}}$$

where m is the size of the lexicon.

3. As a convergence condition, we checked the number of items whose assignments changed in a given iteration. We also limited the number of iterations to 100, although this maximum value was never reached during our experiments.

3.3 Generating the Cluster Representation

After clustering the posts, we use the Kullback-Leibler (KL) divergence between the cluster and the collection in order to select the most informative terms in the cluster:

$$score(t, cluster) = p(t|cluster) \log \frac{p(t|cluster)}{p(t)} \qquad (12)$$

Based on this score, we select the top 100 terms for each cluster as the representation of the cluster. The cluster representation is used as a query, to retrieve the related headlines for that cluster.

3.4 Aggregating the Ranked Lists

After retrieving the headlines for each cluster, we have multiple ranked-lists of headlines per day. By normalizing the scores in each ranked list and aggregating all the ranked lists per day, we will have one headline ranked list per query day. We employ two different approaches for aggregation. In the first approach, we use ComSum method which is a simple summation of the scores in different lists as the final score of the headline [4, 2]. In the second approach we use the Ordered Weighted Averaging (OWA) operators for aggregating scores [9]. The ordered weighted averaging

operator, commonly called the OWA operator, was introduced by Yager [9]. OWA provides a parametrized class of mean type aggregation operators, that can generate an OR operator (Max), an AND operator (Min) and any other aggregation operator in between.

An OWA operator of dimension n is a mapping $F : \mathbb{R}^n \rightarrow \mathbb{R}$ that has an associated weighting vector W,

$$W = [w_1, w_2, ..., w_n]^T$$

such that

$$\sum_{i=1}^{n} w_i = 1, \quad 0 \leq w_i \leq 1,$$

and where

$$F(a_1, ..., a_n) = \sum_{i=1}^{n} w_i b_i \tag{13}$$

where b_i is the ith largest element in the collection $a_1, ..., a_n$. There are different methods for indicating weighting vector W. We use a quantifier based method introduced by Yager [9].

3.5 Classifying the Headlines

Another part of the top stories identification task is to classify headlines into 5 pre-defined classes, namely: *world*, *US*, *sport*, *scitech*, *business*. Since we are not allowed to use any external resource for classification, we decided to use the Reuters corpus itself for classifying the headlines. To this end, we generates 5 short queries for each class manually. Submitting the generated queries to the headlines corpus, we retrieved the top 100 most relevant headlines for each class. We then used the KL divergence to extract the informative terms for each class. Later in order to classify each headline, we calculate its similarity to the 5 class representation and assign it to the most similar class.

4. SESSION TRACK

The session track is held for the first time in TREC 2010. The objective is to take into account the interaction of the user with the search system during a session and not only one single query. User reformulates his query multiple times, in order to clarify what is his information need. To this end, by looking into the results of previous reformulations, search engine will be able to reveal aspects of his information need which were not explicitly stated at first.

4.1 Data Collection

We used the Category B subset of the ClueWeb and indexed it using the Terrier information retrieval system.

4.2 Our Approach

We first generated two ranked lists (RL1 and RL2) for the first and second query using the BM25 implementation of Terrier. We then implemented two different approaches.

1. In the first approach we generate the third ranking (RL3) by scoring documents according to the weighted summation of the reciprocal ranks of documents in RL1 and RL2, where the weight given to documents from RL1 is negative and RL2 is positive. Thus the score for a document d is computed as follows:

$$\texttt{score}(d, \texttt{RL3}) = -\alpha \cdot \frac{1}{\texttt{rank}(d, \texttt{RL1})} + (1+\alpha) \cdot \frac{1}{\texttt{rank}(d, \texttt{RL2})} \tag{14}$$

If a document was not present in one of the ranked lists, it's reciprocal rank was set to 0. We empirically set α to 0.2 and submit a run for this approach.

2. In the second approach we build the relevance model for the first and second query using the top N ranked documents from RL1 and RL2 as the pseudo-relevant documents (denoted PR1 and PR2). We estimate the relevance models R_1 and R_2 by averaging the relative frequencies for terms across the pseudo-relevant documents. We also smoothed the R_1 estimate with a background language model based on collection term frequency estimates as follows:

$$p(w|R_1) = \frac{\lambda}{N} \sum_{d \in \texttt{PR1}} \frac{tf(w,d)}{|d|} + (1-\lambda) \frac{ctf(w)}{\sum_d |d|} \tag{15}$$

$$p(w|R_2) = \frac{1}{N} \sum_{d \in \texttt{PR2}} \frac{tf(w,d)}{|d|} \tag{16}$$

Here $tf(w,d)$ denotes the term frequency of word w in document d, $ctf(w)$ is the collection term frequency and $|d|$ denotes the length of document d.

It is desirable for this task to highlight words from the term distribution of R_2 that are rare in the term distribution of R_1 in order to reduce the number of documents from RL1 that are present in RL3. To this end, we weighted term probabilities in R_2 by their relative information in R_2 and R_1 to calculate a new query model R_3:

$$p(w|R_3) \propto p(w|R_2) \log \frac{p(w|R_2)}{p(w|R_1)} \tag{17}$$

The normalizing constant in this case is simply the Kullback-Leibler divergence between R_2 and R_1. After obtaining the new term distribution, we select the top K terms from R_3 and submit them as a weighted query to Terrier again using the BM25 retrieval function. In this way we generated two runs with the parameters: $N = 10$, $K = 10$ and λ set to either 0.9 or 0.5.

4.3 Experimental Results

Organizers of Session Track released 150 query sessions with three reformulation types: specification, generalization and drifting.

	nsDCG@10	
Run	RL12	RL13
USIML052010	0.2044	0.1855
USIML092010	0.2044	0.1784
USIRR2010	0.2044	0.2044

Table 4: Evaluation scores for the entire session of our three submitted runs using the nsDCG@10 metric

Only 136 sessions (query pairs) were provided with judgement from NIST and were finally evaluated. We submitted

Run	nsDCG_dupes@10 RL12	RL13
USIML052010	0.2069	0.1938
USIML092010	0.2069	0.1869
USIRR2010	0.2069	0.2086

Table 5: Evaluation scores for the entire session of our three submitted runs using the nsDCG_dupes@10 metric

three runs for this task. Runid USIRR2010 belongs to our first approach, while USIML052010 and USIML092010 belong to our second approach where we set λ to 0.5 and 0.9 respectively. Our submitted runs were evaluated by three metrics nsDCG@10, nsDCG_dupes@10, and nDCG@10. The evaluation scores of our three submitted runs with three mentioned metrics are presented in Table 1-3. Table 4 shows the evaluation scores of our systems for task 1 which can be observed by comparing RL1-RL2 and RL1-RL3. Table 5 shows the evaluation scores of our systems for task 2. nsDCG@10 for RL1-RL3 is considered as the official metric for Task 2. We see that USIRR2010 performs better than the two other approaches in both tasks. Table 6 shows the evaluation scores of our submitted runs using the nDCG@10 metric.

Due to the absence of training data (query pairs and corresponding relevance judgements), we were unable to set parameters to maximize performance. Arbitrary setting of the parameters (α and λ) explains the relatively poor performance of the approaches. We intend to test parameter settings on relevance data when it becomes available.

Finally, Table 7 shows the performance of our best run with respect to different formulation types. By comparing RL12 and RL13 it can be seen that this method performs better on "Specification" and "Drifting" as opposed to "Generalization".

Run	nDCG@10 RL1	RL2	RL3
USIML052010	0.1896	0.2144	0.1645
USIML092010	0.1896	0.2144	0.1449
USIRR2010	0.1896	0.2144	0.2147

Table 6: Evaluation scores of our three submitted runs using the nDCG@10 metric

ReformulationType	nsDCG@10 RL12	RL13
Specification	0.1529	0.1532
Drifting	0.1967	0.2013
Generalization	0.2706	0.2652

Table 7: Performance of our best run across the different reformulation types.

5. CONCLUSIONS

We have described our participation in TREC 2010 Blog and Session track for faceted blog distillation, top stories identification and result re-ranking.

For the faceted rankings, we first generated positive and negative facet scores for each retrieved document and then combined the facet rankings with the relevance ranking using Borda Fuse.

For the top stories task we extracted time-stamped blog posts for each query date. Applying a clustering method on the extracted posts for each date, we generated different topics that have been discussed on each day. Then by considering the importance of each extracted topic in the Reuters headline corpus, we generated a ranked list of headline for each day.

In the session track, for re-ranking the documents, we first generated two rank lists for first and second query. We then implemented two different approaches for building the third ranked list. The first method is based on the reciprocal ranks in first two rank lists. In the second approach we build the relevance model for the first and second query using their top ranked documents. We then tried to select terms from the second ranked list which were rare in the first list using the Kullback-Leibler divergence.

6. ACKNOWLEDGMENTS

We thank the TREC organizers for their hard work. We also want to thank Thomson-Reuters for providing us with the headlines corpus.

7. REFERENCES

[1] K. Balog, M. de Rijke, and W. Weerkamp. Bloggers as experts: feed distillation using expert retrieval models. In *Proceedings of SIGIR 2008*, pages 753–754, 2008.

[2] D. Hannah, C. Macdonald, J. Peng, B. He, and I. Ounis. University of Glasgow at TREC 2007: Experiments in Blog and Enterprise Tracks with Terrier. In *Proceedings of TREC*, 2007.

[3] Y. Lee, S.-H. Na, J. Kim, S.-H. Nam, H.-Y. Jung, and J.-H. Lee. Kle at trec 2008 blog track: Blog post and feed retrieval. In *In Proceedings of TREC 2008*, 2008.

[4] C. Macdonald and I. Ounis. Voting for candidates: adapting data fusion techniques for an expert search task. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 387–396. ACM Press New York, NY, USA, 2006.

[5] C. Macdonald, I. Ounis, and I. Soboroff. Overview of the trec-2007 blog track. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC 2007)*, 2007.

[6] C. D. Manning and H. Schtze. *Foundations of Statistical Natural Language Processing*. The MIT Press, June 1999.

[7] J. McQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297, 1967.

[8] I. Ounis, M. De Rijke, C. Macdonald, G. Mishne, and I. Soboroff. Overview of the TREC-2006 blog track. In *Proceedings of TREC*, pages 15–27, 2006.

[9] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Trans. Syst. Man Cybern.*, 18(1):183–190, 1988.