University of Lugano at TREC 2009 Blog Track

Mostafa Keikha, Mark Carman, Robert Gwadera, Shima Gerani, Ilya Markov,
Giacomo Inches, Az Azrinudin Alidin and Fabio Crestani
University of Lugano
Department of Informatics
Lugano, Switzerland
{mostafa.keikha, mark.carman, robert.gwadera, shima.gerani, ilya.markov,
giacomo.inches, az.azrinudin.alidin, fabio.crestani}@usi.ch

ABSTRACT

We report on the University of Lugano's participation in the Blog track of TREC 2009. In particular we describe our system for performing blog distillation, faceted search and top stories identification.

1. INTRODUCTION

Recently, user generated data is growing rapidly and becoming one of the most important source of information in the web. This data has a lot of information to be processed like opinion, experience,etc which can be useful in many applications. Forums, mailing lists, on-line discussions, community question answering sites and social networks like facebook are some of these data resources that have attracted researchers lately.

Blogosphere (the collection of blogs on the web) is one of the main source of information in this category. Millions of people write about their experience and opinion in their blogs everyday, and this provides a huge amount of information to be processed. Due to the importance of this information, TREC (Text REtrieval Conference) has started a new track for blog analysis including opinion detection, polarity mining and blog distillation [7, 11].

In the remainder of this paper we will explain our approach in faceted blog distillation in section 2. Our approach to top stories identification is explained in section 3. We provide conclusions in section 4.

2. BOG DISTILLATION

Blog distillation is the problem of retrieving relevant blogs (as a collection of posts) to a given query. The blog distillation task has been approached from many different points of view. In [3], the authors view it as ad-hoc search and consider each blog as a long document created by concatenating all postings together. Other researchers treat it as the resource ranking problem in federated search [4]. They view the blog search problem as the task of ranking collections of blog posts rather than single documents. A similar approach has been used in [12], where they again consider a blog as a collection of postings and use resource selection approaches. Their intuition is that finding relevant blogs is similar to finding relevant collections in a distributed search environment. In [8], the authors modelled blog distillation as an expert search problem and use a voting model for tackling it.

2.1 Ordered Weighted Averaging Operators in Combining Scores

The ordered weighted averaging operator, commonly called OWA operator, was introduced by Yager [13]. OWA provides a parametrized class of mean type aggregation operators, that can generate OR operator(Max), AND operator(Min) and any other aggregation operator between them.

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

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

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 [13].

OWA operator has different behaviours based on the weighting vector associated with it. Yager introduced two measure for characterizing OWA operator[13]. The first one is called *orness* and is defined as:

$$orness(W) = \frac{1}{n-1} \sum_{i=1}^{n} (n-i) w_i \tag{2}$$

$$orness(W) \in [0, 1]$$

which characterizes the degree to which the operator behaves like an *or* operator. The second measure is *dispersion* and is defined as

$$dispersion(W) = -\sum_{i=1}^{n} w_i \ln(w_i) \tag{3}$$

.

and measures the degree to which OWA operator takes into account all information in the aggregation.

For applying OWA operator to the problem, one important issue is determining weighting vector. Yager introduced a method based on linguistic quantifiers for obtaining this weights:

$$w_i = Q(\frac{i}{n}) - Q(\frac{i-1}{n}), \quad i = 1, 2, ..., n \tag{4}$$

where n is the number of operands to be combined, and Q is the fuzzy linguistic quantifier. We use the following definition for the Q function as suggested by Zadeh[15]:

$$Q(r) = \begin{cases} 0, & \text{if } r < a \\ \frac{r-a}{b-a}, & \text{if } a \leq r \leq b \\ 1, & \text{if } r > b \end{cases} \tag{5}$$

with $a, b, r \in [0, 1]$. We used parameter (a, b) with three different values, $(0, 0.5)$, $(0.3, 0.8)$ and $(0.5, 1)$, as three quantifiers with different levels of orness. Table 1 shows orness and dispersion for each quantifier with values of 5, 10, 20 for n. In this model, n is the number of top relevant posts in each blog that we want to aggregate their relevance score. These relevance scores are calculated by BM25 model in terrier for posts in each blog. Figure 1 and Figure 2 show Mean Average Precision(MAP) and Precision at 10 for experiments over TREC07 datasets. These results reveal that a fixed number of highly relevant posts in each blog is a reliable evidence, using which can result in an effective blog retrieval system.

2.2 Regularizing Relevance Scores

Score regularization is a way of re-calibrating relevance scores for documents based on the relationship between them. The idea behind score regularization is that in accordance with the Clustering Hypothesis, *related documents should have similar scores for the same query.* The authors of [2, 10] propose general models for smoothing document scores based on this hypothesis. In [2], Diaz models the problem in terms of optimization. The goal is to calculate for each document a new (smoothed) score with two contending objectives: score consistency with related documents and score consistency with the initial retrieval score. Diaz defines a cost function $\zeta(f)$ as follows:

$$\begin{aligned} \zeta(f) &= \sigma(f) + \mu\varepsilon(f) \\ &= \sum_{i \neq j} (w_{ij} f_i - w_{ji} f_j)^2 + \mu \sum_i (f_i - y_i)^2 \end{aligned} \tag{6}$$

Here f is a vector of regularized scores over n documents, $\sigma(f)$ is a cost function associated with the inter-document consistency of the scores; if related documents have inconsistent scores, the value of this function will be high. A second cost function $\varepsilon(f)$ measures the consistency with the original scores; if document scores are inconsistent with the original scores, the value of this function will be high. A regularization parameter μ controls the trade off between inter-document smoothing and consistency with the original score vector y. The coefficient w_{ij} in the expansion of $\sigma(f)$ weights the score of the ith document by its similarity to the jth document and is calculated by normalizing (and taking the square root of) values from a symmetric affinity matrix W as follows:

$$w_{ij} = \sqrt{\frac{W_{ij}}{\sum_j W_{ij}}}. \tag{7}$$

Here W_{ij} denotes the similarity between documents i and j. In order to keep the affinity matrix sparse, only the k most similar documents j for each document i have non-zero W_{ij} values[1]. The diagonal values in the matrix W_{ii} are defined to be zero. An iterative solution for the above defined optimization problem is the following:

$$f^{t+1} = (1 - \alpha)y + \alpha \bar{W} f^t \tag{8}$$

Where $\alpha = 1/(1 + \mu)$ is a parameter, $y = f^0$ is the initial score vector, f^t is the score vector after t iterations and $\bar{W}$ is a normalized affinity matrix such that $\bar{W}_{ij} = w_{ij} w_{ji}$. The closed form solution of this problem is given by:

$$f^* = (I - \alpha \bar{W})^{-1} y \tag{9}$$

We used this equation in our experiments. We note that we did not introduce a new model here, but simply investigated the application of graph-based regularization frameworks [2, 10] to the problem of blog distillation, where the aim is not just to rank documents, but to rank blogs which are themselves composed of many documents (posts).

Based on this method we regularize relevance score, which could be the score of the posts or the score of the blog as a whole. In case of posts relevance score we have to aggregate regularized scores again, where we use simple averaging as the aggregation. And in case of regularizing blog score as a whole, we generate one document per blog which is concatenation of its most relevant posts. We use the similarity score of this large document as the blog relevance score and use it on regularization. Table 2 shows the results of posts relevance score regularization over Blog06 dataset with TREC07 and TREC08 query sets.

Table 2: Regularization Results for TREC07 and TREC08 query sets.

Model	MAP	P@10	nDCG	Bpref
TREC07 query sets	0.3126	0.4956	0.5483	0.3118
TREC08 query sets	0.2375	0.3480	0.6990	0.2196

2.3 Faceted Search

For the faceted rankings, we first generated positive and negative facet scores for each retrieved document, denoted $pos(d)$ and $neg(d)$ respectively. These facet scores induce a ranking, denoted $r_{\texttt{pos}}(d, q)$ and $r_{\texttt{neg}}(d, q)$, which we combined with the original relevance ranking $r_{\texttt{rel}}(d, q)$ using the Borda Fuse aggregation method as follows:[2]

$$score_{\texttt{BF}}(d, q) = \alpha \; r_{\texttt{rel}}(d, q) + (1 - \alpha) \; r_{\texttt{facet}}(d, q) \tag{10}$$

Without any training data (i.e. relevance judgments) we were unable to choose an appropriate value for the weighting coefficient α and thus set its value to 0.5.

[1]Some documents may need to have more than k non-zero affinity values in order to keep the matrix symmetric.

[2]Note that whenever there are ties in the ranking, (i.e. documents d_1 and d_2 have the same score), then the rank for those documents is the average of the (total order) ranking.

Table 1: Orness and dispersion for experimented quantifiers in OWA operator

	orness	dispersion	linguistic quantifier
	n=10	n=10	
a=0.0 , b=0.5	0.77	1.609	At least half
a=0.3 , b=0.8	0.44	1.609	Most
a=0.5 , b=1.0	0.22	1.609	As many as possible

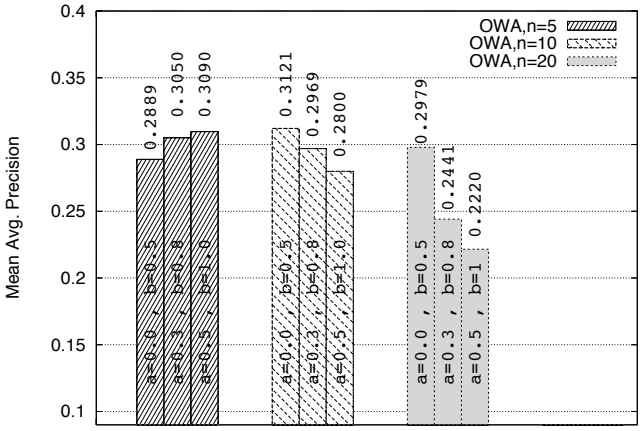


Figure 1: Mean Average Precision

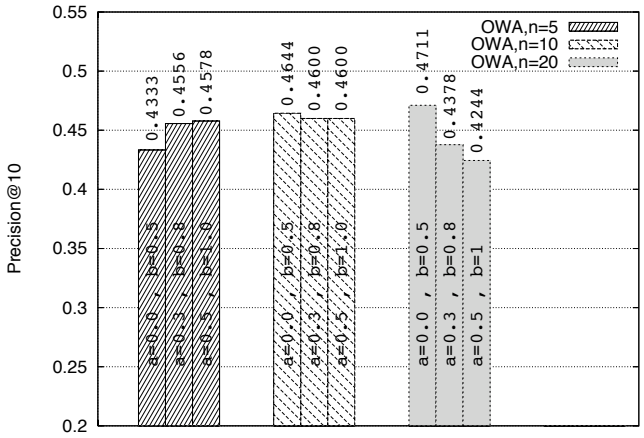


Figure 2: Precision at 10

2.3.1 *In-depth versus Shallow*

For the in-depth versus shallow facet, we calculated the Cross Entropy (CE) between each retrieved document 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:

$$pos(d) = CE(p(.|d), p(.|c)) = \sum_{t \in d} p(t|d) \log \frac{1}{p(t|c)} \qquad (11)$$

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: $p(t|d) = \texttt{tf}(t,d)/\sum_{t'} \texttt{tf}(t',d)$, where $\texttt{tf}(t,d)$ is the absolute term frequency. Meanwhile $p(t|c)$ denotes is the probability of a term across the whole collection c, for which we use a document frequency based estimate $p(t|c) = \texttt{df}(t)/|c|$ 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. $neg(d) = -pos(d)$.

2.3.2 *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 [9]. 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 14 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:

$$pos(t) = p(\mathcal{T}, \mathcal{O}) \log \frac{p(\mathcal{T}, \mathcal{O})}{p(\mathcal{T}), p(\mathcal{O})} + p(\bar{\mathcal{T}}, \bar{\mathcal{O}}) \log \frac{p(\bar{\mathcal{T}}, \bar{\mathcal{O}})}{p(\bar{\mathcal{T}}), p(\bar{\mathcal{O}})} \qquad (12)$$

The negative facet score is calculated analogously as follows:

$$neg(t) = p(\mathcal{T}, \bar{\mathcal{O}}) \log \frac{p(\mathcal{T}, \bar{\mathcal{O}})}{p(\mathcal{T}), p(\bar{\mathcal{O}})} + p(\bar{\mathcal{T}}, \mathcal{O}) \log \frac{p(\bar{\mathcal{T}}, \mathcal{O})}{p(\bar{\mathcal{T}}), p(\mathcal{O})} \qquad (13)$$

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:

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

Where $\texttt{df}(t, O)$ is the number of opinionated documents containing the term t. The other joint and marginal probabilities required for equations 12 and 13 are estimated analo-

gously.

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:

$$pos(d) = E_d[pos(t)] = \sum_{t \in d} p(t|d)pos(t) \qquad (14)$$

2.3.3 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".

3. TOP STORIES IDENTIFICATION

Our method for the top stories task proceeded as follows. We first extracted time-stamped news stories for each query date while filtering out non-news related items. For each query date we also extracted the set of blog posts that were posted on the same or following days and where the post had some vocabulary overlap with corresponding set of news stories. Each set of blog posts was then clustered using an incremental clustering algorithm. Next we ranked clusters with respect to size and time-span in order to identify the most important clusters pertaining to the corresponding news stories. Finally we identified the most authoritative document for the 10 most important clusters on each query date.

In the following sections we outline our approach in more detail.

3.1 The algorithm

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

1. for every query date we extract a set of time-stamped news stories by using the date part of the permanent links of the urls and we filter out non-news related documents

2. for every query date we extract a set of time-stamped blog posts that satisfy the following condtions: (I) they were posted on the same day and a following three days and (II) they have a vocabulary overlap with the corresponding time-stamped sets of news stories.

3. we cluster every time-stamped blog-post set using an incremental clustering algorithm whose details are presented in Section 3.2.

4. we identify the most important clusters pertaining to the corresponding time-stamped news stories by ranking clusters with respect to size and time-span.

5. we filter out clusters that correspond to the news-stories by using the centroid score.

6. we identify the most authoritative document for the top-10 most important clusters for every query date using the ranking algorithm presented in Section 3.3.

Table 3: Example documents for which the triangle inequality does not hold

doc	w_1	w_2	w_3	w_4	w_5	w_6
d_1	1	1	0	0	0	0
d_2	0	1	1	1	1	0
d_3	0	0	0	0	1	1

3.2 Incremental document clustering in sliding time-window

In this section we present our implementation of an incremental variant of a non-hierarchical document clustering algorithm using a similarity measure based on nearest neighbors (NN-based) [5, 6, 1].

We preprocess every document using the following steps: (I) *HTML parsing*; (II) *tokenization*; (III) *stemming*; and (IV) *stopwords removal*. We represent a document d using the term vector model where $d = [w_1, w_2, \ldots, w_d]$ and w_i is the weight of the the i-th term (word) that was extracted from the document after the preprocessing of the original document. The reason we use a NN-based similarity measure between news stories is because direct similarity measures between two vectors like *Euclidean distance* and the *dot product* have the following problems in a high dimensional space: there is an experimental evidence that they are not reliable [5] and the triangle inequality does not hold. For an example where the triangle inequality does not hold consider the following three vectors representing hypothetical documents: d_1, d_2 and d_3 over six terms in Table 3. Thus, although d_1 is close to d_2 by sharing one term and d_2 is close to d_3 by also sharing one term d_1 and d_3 do not share any terms. There are the following reasons why the triangle inequality does not hold for documents: (I) diversity of term usage to express the same meaning with respect to the same event, which is aggravated by the fact that we consider similarity between documents across different news sources; and (II) content of stories reporting the same event may change throughout time and may use a different vocabulary. Therefore the clusters containing the documents are inherently non globular justifying the use of the NN-based versus a centroid-based similarity measure.

We perform the standard TF-IDF weighting of the document term vector d: $w_i = tf_i \cdot idf_i$, where: tf_i is within document *term frequency* of term t_i and $idf_i = \log(N/df_i)$ is the *inverse document frequency*, where N is the total number of documents in the collection and df_i is the *document frequency* of term t_i defined as the number of documents containing the term in the collection.

In order to present the clustering algorithm we introduce the following notation. Let $sim(d_i, d_j) = \frac{\sum_{k=1}^{n} d_{i_k} \cdot d_{j_k}}{\|d_i\| \cdot \|d_j\|}$ be the *cosine similarity* or *content similarity* between documents d_i and d_j, where $sim(d_i, d_j) \in [0, 1]$. Let $\mathcal{N}_{\tau_d}(d_i)$ be the neighborhood of d_i defined as a set of documents for which $sim(d_i, d) \geq \tau_d$, where $d \in \mathcal{N}_{\tau_d}(d_i)$. Let $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ be the set of active clusters in the window. Let $\mathcal{C}(\mathcal{N}_{\tau_d}(d_i)) \subseteq \mathcal{C}$ be the set of clusters that contain any documents in $\mathcal{N}_{\tau_d}(d_i)$. Let $\mathcal{N}_{\tau_d}(C_j, d_i)$ be the subsets of documents in $\mathcal{N}_{\tau_d}(d_i)$ belonging to cluster $C_j \in \mathcal{C}$ such that $\mathcal{N}_{\tau_d}(C_j, d_i) = \emptyset$ if cluster C_j has no members in $\mathcal{N}_{\tau_d}(d_i)$. Let $\Delta(d_i, C) = \sum_{d \in C} sim(d_i, d)$ be the similarity between

d_i and the set of documents $d \in C$. Let $df^{(i)}$ be the document frequency vector for stream i. Let $currentTime$ be the timestamp of the most recent document in the window, i.e, the current timestamp of the window.

The clustering algorithm proceeds as follows:

1. **Neighborhood search**: given a new document d_i identify its neighborhood $\mathcal{N}_{\tau_d}(d_i)$

2. **Identification of a cluster that can accept a new document**: For every cluster $C \in \mathcal{C}(\mathcal{N}_{\tau_d}(d_i))$ compute $\Delta(d_i, \mathcal{N}_{\tau_d}(C, d_i))$. Select a cluster

$$C_{max} = \max_{C \in \mathcal{C}(\mathcal{N}_T(C,d_i))} \Delta(d_i, \mathcal{N}_T(C, d_i)).$$

   If $\mathcal{N}_{\tau_d}(d_i)$ is empty then create a new cluster C_{new} for d_i.

3. **Merging**: merge every set $C \in \mathcal{C}(\mathcal{N}_{\tau_d}(d_i)) \setminus C_{max}$ with C_{max}.

For achieving an efficient neighborhood search in the window we dynamically maintain an inverted index data structure in the time-window. Also we maintain an independent document frequency vector $df^{(i)}$ for each stream i in order to suppress terms whose popularity is specific to a particular news source.

The sliding window process proceeds as follows. When a new document $d_t^{(i)}$ arrives the following actions are executed: (I) the document is added to the window, which involves adding the corresponding terms to: the inverted index and $df^{(i)}$ vector; (II) $d_t^{(i)}$ is clustered using the presented algorithm and if the result is a singleton cluster then it is added to the set of active clusters $\mathcal{C}$; (III) $currentTime$ is set to $d_t^{(i)}.timestamp$; (IV) documents which are older than $currentTime - w$ are removed from the window, which involves removing corresponding entries in: the set of active clusters $\mathcal{C}$ and the inverted index

Thus the presented clustering algorithm has the following parameters: (I) the time-window size $w = 24$ hours; (II) the document similarity threshold $\tau_d = 0.5$. Our evaluation of the clustering results suggest that $Precision = 95\%$. We selected $\tau_d = 0.5$ based on an experimental evaluation that showed $\tau_d = 0.5$ to be a good compromise with respect to precision and recall.

3.3 Content-aware ranking function

In this section we present a content-aware ranking function that ranks with respect to the following factors:

1. the importance of a cluster increases with its size and decreases with its time-span (the time distance between the first and the last document)

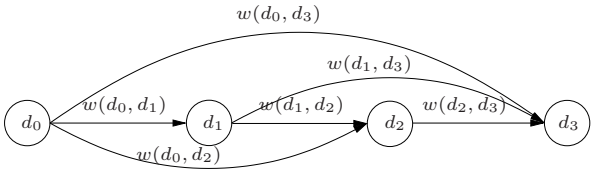
2. the importance of a document in a given position (its authority) in a time-ordered cluster is proportional to the difference between the average combined similarity (content similarity and temporal distance) for the following documents and the previous documents in the cluster.

The first factor is an extension of the first factor for the probabilistic ranking function by prioritizing clusters that are proximate in time. This corresponds to the fact that a large cluster on a given event that is proximate in time means that the event is very important since every source reports it in a very short time window.

The second factor has the following motivation. It is known that news stories discussing the same event tend to be temporally proximate across the news streams [14]. Therefore we use a combine similarity measure that increases with the content similarity and decreases with the temporal distance. Let $\Delta_t(i,j)$ be the temporal distance between documents d_i and d_j, where $\Delta_t(i,j) = e^{-\alpha(d_i.time - d_j.time)}$ and $\alpha = -\frac{ln(dFactor)}{w}$, where $dFactor$ is the decaying factor that denotes the factor by which the value of the function decays within the time interval w being the time window size. Then the *combined similarity* $w(d_i, d_j)$, can be expressed as follows

$$w(d_i, d_j) = sim(i,j) \cdot \Delta_t(i,j), \tag{15}$$

where $sim(i,j)$ is the content similarity. Figure 3 presents a graphical representation of the dependencies between documents in a cluster with respect to the combined similarity, where a directed edge from an earlier to a more recent document has a weight equal to the combine similarity.



$w(d_i, d_j) = sim(i,j)\Delta_t(i,j)$

$Prev(s, d_1.steam, 1) = \{d_0\}$ $Follow(s, d_1.steam, 1) = \{d_2, d_3\}$

$In(s, d_1.steam, 1) = w(0,1)$ $Out(s, d_1.steam, 1) = \frac{w(1,2)+w(1,3)}{2}$

$authority(s, d_1.stream, 1) = \frac{w(1,2)+w(1,3)}{2} - w(0,1)$

Figure 3: Combined similarity between documents in itemset-sequence (cluster) $s = [d_1, d_2, d_3, d_4]$. $Prev(s, d_1.stream, 1)$ **and** $Follow(s, d_1.stream, 1)$ **are sets of documents preceding and following document** d_1 **in position** 1. $In(s, d_1.stream, 1)$ **and** $Out(s, d_2.stream, 1)$ **are the average combined similarity for** $Prev(s, d_1.stream, 1)$ **and** $Follow(s, d_1.stream, 1)$ **respectively.** $authority(s, d_1.stream, 1)$ **is the authority of source** $d_1.stream$ **in position** 1.

Now we define the average combined similarity with respect to previous and following documents. Given an itemset-sequence $s \in \mathbf{S}$ we define the following two sets. Let $Prev(s,i,j) = \bigcup_{l=1, l \leq j} d_l$ be the set of documents that precede document $d^{(i)}.stream = i$ in position j in s. Let $Follow(s,i,j) = \bigcup_{l=j+1, l \leq |s|} d_l$ be the set of documents that follow document $d^{(i)}.stream = i$ in position j in s. Then the average combined similarity with respect to previous documents (in positions $j' < j$), denoted $In(s,i,j)$, can be expressed as follows

$$In(s,i,j) = \frac{1}{|Prev(s,i,j)|} \sum_{d \in Prev(s,i,j)} w(d, d_j). \tag{16}$$

Also the average combined similarity with respect to the following documents (in positions $j < j'$), denoted $Out(s,i,j)$,

can be expressed as follows

$$Out(s,i,j) = \frac{1}{|Follow(s,i,j)|} \sum_{d \in Follow(s,i,j)} w(d,d_j). \quad (17)$$

Given the value of $In(s,i,j)$ and $Out(s,i,j)$ we define "authority" of source i corresponding to a document in position j as follows

$$authority(s,i,j) = Out(s,i,j) - In(s,i,j). \quad (18)$$

This measure of authoritativeness prioritizes sources that: (I) "borrow" little content form previous documents ($In(s,i,j)$) and whose content is widely "borrowed" by following documents in the cluster ($Out(s,i,j)$) and (II) produce a timely content ($|Follow(s,i,j)|$ is the biggest equal to $|s|-1$ and $|Prev(s,i,j)|$ is the smallest equal to 0 for the first story in the cluster ($j=0$)). This measure of authoritativeness has many desired properties. For example consider a case where there is source i_2, which always follows an authoritative source i_1 with very similar content. Then $authority(s,i_2,j)$ will be very small (even negative) for i_2 since it only "repeats" the content of i_1. Thus, this case may correspond to a reuse of content by i_2 from i_1, where i_2 repeats content from i_1 within a short time window. In other words (18) discriminates between "producers" of the content (positive value of (18)) and "repeaters" (negative value of (18)). However, note that because of limitations of the cosine similarity measure we are unable to decide with hundred percent confidence that one story is reusing content from another one.

We now define the rank of a cluster s as follows

$$rankCluster(s) = w_{cluster}(k) \cdot \Delta_t(0,|s|-1) \quad (19)$$

where $w_{cluster}(k)$ is the weight of the cluster of size k (size of the cluster) and $\Delta_t(0,|s|-1)$ is the time-span of the cluster.

Despite the sophisticated clustering machinery used in the top stories identification, our results were poor due to the fact that we were only able to run the clustering over a small subset (around 10%) of the data. It was mainly because of the time restriction and the computational load required by the algorithm on the very large dataset.

4. CONCLUSIONS

We have described our participation in TREC 2009 Blog track for faceted blog distillation and top stories. We implemented two types of algorithms for blog distillation. In one of our experiments, we used fuzzy aggregation methods for combining post relevance scores in each blog to calculate blog scores as a whole. In another part of the experiments, we used regularization methods for smoothing relevance scores based on the similarity between the retrieved blogs. We carried out regularization on two types of scores: posts relevance scores and large document relevance scores (where each blog is represented by the concatenation of its most relevant posts). Finally we combined the two methods (regularization and OWA) to take into account the similarity between retrieved posts while performing good aggregation over them, to generate new scores for each blog.

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 top stories task we first extracted time-stamped news stories for each query date while filtering out non-news related items. For each query date we also extracted the set of blog posts that were posted on the same or following days and where the post had some vocabulary overlap with corresponding set of news stories. Each set of blog posts was then clustered using an incremental clustering algorithm. Next we ranked clusters with respect to size and time-span in order to identify the most important clusters pertaining to the corresponding news stories. Finally we identified the most authoritative document for the 10 most important clusters on each query date.

5. REFERENCES

[1] S. Chung and D. McLeod. Dynamic pattern mining: An incremental data clustering approach. *Journal on Data Semantics, Lecture Notes in Computer Science*, 2:85–112, 2005.

[2] F. Diaz. Regularizing ad hoc retrieval scores. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 672–679, 2005.

[3] M. Efron, D. Turnbull, and C. Ovalle. University of Texas School of Information at TREC 2007. In *Proc. of the 2007 Text Retrieval Conf*, 2007.

[4] J. L. Elsas, J. Arguello, J. Callan, and J. G. Carbonell. Retrieval and feedback models for blog feed search. In *SIGIR*, pages 347–354, 2008.

[5] L. Ertöz, M. Steinbach, and V. Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *in Proceedings of Second SIAM International Conference on Data Mining*, 2003.

[6] R. A. Jarvis and E. A. Patrick. Clustering using a similarity measure based on shared near neighbors. *IEEE Transactions on Computers*, 22(11):1025–1034, 1973.

[7] 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.

[8] 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.

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

[10] Q. Mei, D. Zhang, and C. Zhai. A general optimization framework for smoothing language models on graph structures. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 611–618. ACM, 2008.

[11] 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.

[12] J. Seo and W. Croft. UMass at TREC 2007 Blog Distillation Task. In *Proc. of the 2007 Text Retrieval Conf*, 2007.

[13] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making.

IEEE Trans. Syst. Man Cybern., 18(1):183–190, 1988.

[14] Y. Yang, J. G. Carbonell, R. D. Brown, T. Pierce, B. T. Archibald, and X. Liu. Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, 14(4):32–43, 1999.

[15] L. Zadeh. A computational approach to fuzzy quantifiers in natural languages. *International series in modern applied mathematics and computer science*, 5:149–184, 1983.