Linguistic Aggregation Methods in Blog Retrieval

Mostafa Keikha and Fabio Crestani
University of Lugano
Lugano, Switzerland
{mostafa.keikha,fabio.crestani}@usi.ch

July 26, 2010

Abstract

This paper addresses the blog distillation problem, that is, given a user query find the blogs that are most related to the query topic. We model each post as evidence of the relevance of a blog to the query, and use aggregation methods like Ordered Weighted Averaging (OWA) operators to combine the evidence. We show that using only highly relevant evidence (posts) for each blog can result in an effective retrieval system. We also take into account the importance of the posts in a query-based cluster and investigate its effect in the aggregation results. We use prioritized OWA operators and show that considering the importance is effective when the number of aggregated posts from each blog is high. We carry out our experiments on three different data sets (TREC07, TREC08 and TREC09) and show statistically significant improvements over state of the art model called Voting Model.

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 (Macdonald, Ounis, & Soboroff, 2007; Ounis, De Rijke, Macdonald, Mishne, & Soboroff, 2006).

In this paper we focus on the blog distillation task which is: given a user query find the blogs that are most related to the query topic. In other words, we want to find the most important blogs for a specific topic which by reading them, user will have as much information as possible.

There are some properties of blogs that make blog analysis different from usual text analysis. One of these properties is related to the time stamp assigned to each post; it is possible that the topics of a blog change over time and this can affect blog relevance to the query. For example a blog which has more relevant posts lately would be more relevant to the query than a blog with the same number of relevant posts that are older. Also each post in a blog can have viewer generated comments that can change the relevance of

the blog to the query if these are considered as part of the content of the blog. Another property is related to the meaning of the links between blogs which are different than links between websites. Links in the blogosphere show content relation between the source and the destination that can be similarity or agreement or disagreement about a topic, while links between websites are more about trust and can be used as a measure of authority of the destination (Kleinberg, 1999). Finally, blog distillation is different from traditional ad-hoc search since the retrieval unit is a blog (a collection of posts), instead of a single document. With this view, blog distillation is similar to the task of resource selection in federated search (Elsas, Arguello, Callan, & Carbonell, 2008). In this paper we focus on this last mentioned property and try to use aggregation methods like Ordered Weighted Averaging (OWA) operators to combine post relevance and compute blog relevance as a whole.

The rest of the paper is organized as follows: in section 2 we review related work, specially voting models as the state of the art fusion methods applied to blog distillation. Section 3 includes description of the Ordered Weighted Averaging operators as the aggregation methods implemented in our experiments. Section 4 explains a graph based method for calculating the importance of posts in a query-based cluster, which we involve it later in the aggregation. Section 5 shows our experimental results and section 6 gives conclusions and future work.

2 Related Work

The main research in blog distillation started after 2007, when TREC organizers proposed this task in the blog track. Researchers have applied different methods from similar problems to blog distillation like ad-hoc search methods, expert search algorithms or methods from resource selection in distributed information retrieval.

In (Efron, Turnbull, & Ovalle, 2007), authors use ad-hoc search methods for finding relevant blogs to a specific topic, where they treat each blog as one long document created by the concatenation of all its posts. Given a query q they derive a score for each blog b in the corpus using the negative KL-divergence between the query language model and the language model of b as a whole. Authors in (Nunes, Ribeiro, & David, 2008) use temporal evidence as an extra feature of blogosphere beside content of the blogs. They use temporal span and temporal dispersion as two measures of relevance over time and show that these features can help in blog retrieval.

Expert Search is a task in TREC Enterprise Track where systems are asked to rank candidate experts with respect to their predicted expertise about a query, using documentary evidence of the expertise found in the collection (Soboroff, de Vries, & Craswell, 2006). In (Macdonald & Ounis, 2006b), authors create a voting model for expert search and later they use this model for blog search (Hannah, Macdonald, Peng, He, & Ounis, 2007). In this model, each post in a blog has be seen as a weighted vote for that blog

to have an interest in the query topic. Then using data fusion models they combine these votes to compute the final relevance score of the blog. They also try to use other features like cohesiveness (on average, how different each post is from the blog as a whole) and anchor text similarity to the query(the anchor text of links that point to the blog). However, their experiments show that these features do not improve performance of the system.

Resource selection in distributed information retrieval is another similar problem and some of its methods are applied to blog distillation. In distributed information retrieval, because searching all servers for each query is so expensive, some server selection algorithms should be used (Hawking & Thomas, 2005). Queries are sent to servers that have more relevant documents to the query. Authors in (Elsas et al., 2008; Arguello, Elsas, Callan, & Carbonell, 2008) deal with blog distillation as a recourse selection problem. They model each blog as a collection of posts and use a language modelling approach to select the best collection. Similar work has been introduced in (Seo & Croft, 2008), called Pseudo Cluster Selection, where they create topic-based clusters of posts in each blog and select blogs which have the most similar clusters to the query.

Our work in this paper is similar to resource selection algorithms for distributed information retrieval, where we want to use aggregation methods to combine available evidence and find the most relevant blogs. The intuition in this paper is mostly inspired by work in (Lee, Na, Kim, Nam, Jung, & Lee, 2008), which says "a few posts that are highly relevant to a given topic

represent the blog feed relevance". We try to show that fixed number of highly relevant posts in each blog is a good indicator of relevance of that blog to the query. So we do not need to use all the posts in the blog to find if it is relevant or not. It makes the retrieving process faster, and the experimental results show that it works well on real data and improves on the state of the art methods.

As the baseline, we use voting models (Macdonald & Ounis, 2006b; Hannah et al., 2007; Macdonald & Ounis, 2008) to aggregate relevance score of posts for each blog. In these models, authors use expert search methods in blog distillation and treat blogs as experts. The idea is that the blog distillation task can be seen as a voting process: A blogger with an interest in a topic will blog regularly about the topic, and these blog posts will be retrieved in response to a query topic. Each time a blog post is retrieved in response to a query topic, that can be seen as a vote for that blog to have an interest in the topic area. Authors use fusion methods to find related blogs. These methods rank blogs by aggregating the relevance scores of the posts associated with each blog. We use their best two aggregation methods, called expCombSum and expCombMNZ, as the baselines. ExpCombSum ranks blogs by considering the sum of the exponential of the posts relevance score as follows:

$$score_{expCombSum}(B,Q) \;=\; \sum_{p \in R(Q) \cap Posts(B)} exp(score(p,Q)) \qquad (1)$$

where B is a blog, $posts(B)$ indicates posts in blog B, $R(Q)$ is the set of posts retrieved for query Q and score(p,Q) is the similarity between the post and the query that is computed using a search engine. ExpCombMNZ includes a component which takes into account the number of posts in each blog that are in the ranked list of the query, *R(Q)*:

$$score_{expCombMNZ}(B,Q) \;=\; ||\, R(Q) \cap posts(B) ||\, . \sum_{p \in R(Q) \cap Posts(B)} exp(score(p,Q))$$

where ||.|| shows the size of the set. For more details about voting models in blog retrieval refer to (Macdonald & Ounis, 2006b; Hannah et al., 2007; Macdonald & Ounis, 2008).

3 Ordered Weighted Averaging Operators

The Ordered Weighted Averaging operators, commonly called OWA operators, were introduced by Yager (Yager, 1988). OWA operators provide a parametrized class of mean type aggregation operators, that can generate OR operator (Max), AND operator (Min) and any other aggregation op-

erator between them.

An OWA operator of dimension n is a mapping $F : \mathbb{R}^n \to \mathbb{R}$ 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

$$OWA(a_1, ..., a_n) = \sum_{i=1}^{n} w_i a_{ind(i)} \tag{2}$$

where $a_{ind(i)}$ is the ith largest element in the operand collection $a_1, ..., a_n$.

As can be seen, when $w_1 = 1$, the operator returns the largest element (like an OR operator) and when $w_n = 1$, it generates the smallest element (like an AND operator). By changing the weighting vector, we can have any operator between these two extremes. For example, when $\forall i; w_i = 1/n$, it will be a mean operator.

OWA operators have different behaviours based on the weighting vector associated with them. Yager introduced two measures for characterizing OWA operator (Yager, 1988). The first one is called *orness* and is defined as:

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

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

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

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

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

As mentioned before, one of the issues in blog retrieval is aggregating posts relevance score and calculate a score for the blog as a whole. Since relevance of a post to a query is a fuzzy property, each post can be seen as a source of information for the blog being relevant to the query and fuzzy aggregation methods like OWA operators seem applicable in blog retrieval.

3.1 Linguistic-functional Specification for Weighting Vector

One important issue in applying OWA operators to a problem is determining the weighting vector. Yager introduced a method based on linguistic quantifiers for obtaining these weights:

$$w_i = Q\left(\frac{i}{n}\right) - Q\left(\frac{i-1}{n}\right), \quad i = 1, 2, ..., n \tag{3}$$

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 (Zadeh, 1983):

$$Q(r) = \begin{cases} 0, & \text{if } r < a \\ \dfrac{r-a}{b-a}, & \text{if } a \leq r \leq b \\ 1, & \text{if } r > b \end{cases} \tag{4}$$

with $a, b, r \in [0, 1]$. With different values for a and b, we can define different linguistic quantifiers like *"Most"*, *"At least half"* and *"As many as possible"*. We use these quantifiers for computing the weighting vector in our experiments.

3.2 Importance weighted OWA aggregation

As described in (Yager, 2009), it is possible to include importance associated with each operand in the aggregation result. In this way we can take into account other available information in the final aggregated score. In our problem, the importance could be calculated based on the hyper-link information, user comments, temporal information or content relations between posts.

In this case, beside the relevance score for each post, we are given the importance values for them also. So the final aggregated score for a blog, $Score(B)$, will be:

$$Score(B) = OWA((a_1, v_1), (a_2, v_2), \ldots, (a_n, v_n)) \tag{5}$$

where (a_i, v_i) tuple is the given information for each blog post with a_i as the relevance score of the post and v_i as its importance. Again we assume $a_{ind(j)}$ as the jth largest element of a_i and $v_{ind(j)}$ as its associated importance weight. By defining $R = \sum_{j=1}^{n} v_{ind(j)}$ as the sum of the importance weights, the OWA weights are calculated by:

$$u_j = Q\left(\frac{R_j}{R}\right) - Q\left(\frac{R_{j-1}}{R}\right) \tag{6}$$

where $R_j = \sum_{k=1}^{j} v_{ind(k)}$ and $R_0 = 0$. We use the same linguistic-based functions, as described in section 3.1, as our scope function Q. After calculating the weighting vector, the aggregated score for each blog is obtained by:

$$Score(B) = OWA((a_1, v_1), (a_2, v_2), \ldots, (a_n, v_n)) = \sum_{j=1}^{n} u_j a_{ind(j)} \tag{7}$$

As we can see in equation 6, when the importance is zero, the weight for that operand will be zero and it will be ignored:

$$v_i = 0 \Rightarrow R_i = R_{i-1} \Rightarrow u_i = 0 \tag{8}$$

Similarly when the importance is very small, the difference between R_i and R_{i-1} will be very small and the u_i will be small consequently. However when $R_i > R_j$, there is no guarantee that $u_i > u_j$, because we rank the operands by relevance scores a_i. If we use importance values v_i for both ranking and weight calculation, the operator will be same as the Importance Induced

Ordered Weighted Averaging (I-IOWA) as introduced in (Chiclana, Herrera-Viedma, Herrera, & Alonso, 2007). I-IOWA guarantees higher weights by increasing importance, however because special setting in our application we want to give more priority to the relevance scores than importance values and we do not want this property. We simply want to smooth the relevance score based on the importance of the document.

4 Posts importance in a query-based cluster

In order to integrate importance in the OWA operators based on the Formula 7, we need to define a measure of importance for each blog post. We use the content-based relation between the documents to calculate their importance. We consider a set of retrieved documents for each query as a cluster of relevant documents for that query. The more a document is similar to the other retrieved documents, more important it is in the cluster.

So we can use the similarity between retrieved documents as an importance measure. To capture these similarities in a general way, we define a graph based representation of the cluster which includes the retrieved documents for a topic and the terms occurring in those documents. Figure 1 shows part of such a graph where an edge between a post and a term indicates that the term occurred in that specific post. The more common terms there are between two documents, the more similar those documents should be. By performing a long enough random walk in this matrix and reaching

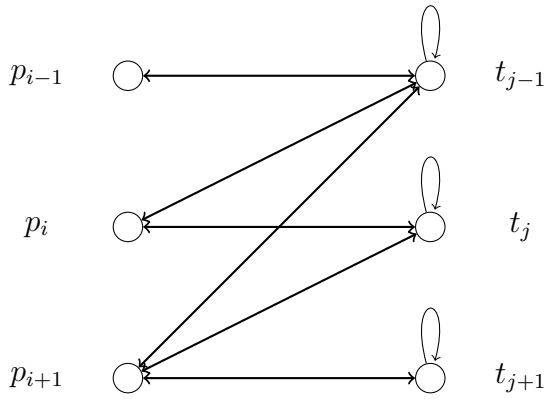

Figure 1: Example of a post-term graph

the stationary distribution, we will have the importance of each individual document in the graph.

We represent the graph by its adjacency matrix A where each row is equivalent to a node in the graph and elements of the matrix show the edges of the graph. In this matrix A_{ij} (element of row i and column j in the matrix A) denotes the transition probability from node i to node j in one step of a random walk, i.e. $P(t_j|p_i) = A_{ij}$ shows the probability of a term for a given post (transition probability from a post to a term) and $P(p_i|t_j) = A_{ji}$ shows the probability of a post for a given term (transition probability from a term to a post). The outgoing probabilities from a node and hence the values in any row of A add up to one, i.e. $\sum_j A_{ij} = 1$. It is worth noting that the matrix A will be a square matrix that number of rows (columns) is equal to the total number of posts and terms.

We compute the transition probabilities between nodes in more than one

step by multiplying the matrix A by itself. Element C_{ij}, where $C = A^n$, is thus the probability of reaching j from i after n steps in a random walk.

To make the computation less expensive and calculate the importance around the query, we just compute the stationary probability for reaching to the query terms from each document, instead of calculating the probabilities for all the terms. We will use the notation $P_n(t_j|p_i) = (A^n)_{ij}$ to denote the probability of moving from post p_i to query term t_j in n steps. Performing this random walk can be seen as a type of smoothing where we compute probability estimates even for terms not present in the document. This graph-based smoothing takes into account the frequency of each term in similar documents, where similar documents are determined by their common terms. For example, even if a particular blog post discussing some aspect of machine learning did not contain the word "regularization", the term would be assigned a non-zero value within a smoothed term probability distribution for the post, since other posts discussing machine learning would likely contain this term with high frequency.

Since we only want to compute $P_n(t_j|p_i)$ for terms in the query, we can efficiently calculate this value, as indicated in (Craswell & Szummer, 2007), by iterating forward from *each* query term node t_j to *all* post nodes p_i concurrently, as follows:

$$P_n(t_j|p_i) = (A^n)_{ij} = (A(...(A(A\mathbf{j})))_i \tag{9}$$

Where $\mathbf{j}$ is a unit (column) vector with value 1 in the j-th row. Note that this calculation only needs to be performed once per query term. The transition probabilities between terms and documents are calculated using Maximum Likelihood estimate:

$$P(t_j|p_i) = \frac{\text{tf}(t_j, p_i)}{|p_i|} = \frac{\text{tf}(t_j, p_i)}{\sum_k \text{tf}(t_k, p_i)} \tag{10}$$

Where $\text{tf}(t_j, p_i)$ is the frequency of the term t_j in the post p_i and $|p_i|$ shows size of the post which is the total number of terms in it. Similarly, the probability of transitioning from a term node t_j to a post node p_i is defined using Bayesian theorem by:

$$P(p_i|t_j) = \frac{\text{tf}(t_j, p_i)}{\sum_k \text{tf}(t_j, p_k)} \tag{11}$$

This formula can be concluded directly from equation 10 by assigning proper prior probabilities for posts and terms. We set the prior probability of a post to be proportional to its size:

$$p(p_i) = \frac{|p_i|}{\sum_k |p_k|} \tag{12}$$

and probability of a term is set to be proportional to its frequency in the collection:

$$p(t_j) = \frac{\sum_k \text{tf}(t_j, p_k)}{\sum_k |p_k|} \tag{13}$$

Using these prior probabilities in Bayesian theorem equations 10 and 11 can be derived from each other.

A random walk of length one is equivalent to a Maximum Likelihood estimate for $P_n(t_j|p_i)$, while an infinite random walk would generate a stationary probability distribution independent of the starting points.

By adding a self-loop transition to all term nodes, we turn a length n random walk into the weighted (exponentially decaying) average of walks of length 1 to n. We do this by $\alpha = P(t_j|t_j)$ as the "self-loop" probability on the term nodes. It is a smoothing parameter which regulates the importance of shorter versus longer walks in the post-term graph. Thus the parameter α regulates how much smoothing we do on the initial Maximum Likelihood estimate. The smaller the self-loop probability α, the more the estimate will be smoothed with longer walks and vice versa.

To see the effect of α in the final probabilities in the random walk, assume we calculate a walk of length 20 in a graph. At the end of this calculation, we have $P_{20}(t_j|p_i)$ for all terms and posts. However the calculated probabilities do not have any information about probabilities in short walks and we miss those information. E.g it is possible that all the paths between a post and a term have odd length, then the P_{20} for that post and term will be zero, while a shorter path with length 19 might have a value. By adding the self loop on the terms, we keep a history of all walks to that term.

At the end, to make the matrix stochastic (sum up to one in each row) by having term self loops, we decrease the probabilities from terms to posts.

Linguistic quantifier	n	orness	dispersion
At least half (a=0.0, b=0.5)	5	0.79	1.054
	10	0.77	1.609
	20	0.77	2.302
	30	0.76	2.708
Most (a=0.3, b=0.8)	5	0.44	1.054
	10	0.44	1.609
	20	0.44	2.302
	30	0.44	2.708
As many as possible (a=0.5, b=1.0)	5	0.21	1.054
	10	0.22	1.609
	20	0.23	2.302
	30	0.24	2.708

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

The generated matrix looks like:

$$A = \begin{bmatrix} 0 & M_{PT} \\ (1-\alpha)M_{TP} & \alpha I \end{bmatrix} \tag{14}$$

where P indicates the posts, T shows the terms and M_{XY} shows a stochastic sub-matrix with transition probabilities from the object type X to the object type Y. Once we have computed $P_n(t_j|p_i)$ for each term in the query we can use these values (after further smoothing with a collection model $P(t_j|C)$) to calculate an estimate for the query likelihood given the post and use it as the post importance in the cluster around the query:

$$importance(p_i, Q) = P_{RW}(Q|p_i) = \prod_{t_j \in Q} \lambda P_n(t_j|p_i) + (1-\lambda)P(t_j|C) \tag{15}$$

We use this importance value, as an another information available for the posts, in the Formula 7 and aggregate blog post relevance scores and calculate blog relevance score as a whole.

5 Experimental results

For evaluating our methods we use three years worth of TREC blog track data that is the set of available data sets for the blog distillation task including TREC'07, TREC'08 and TREC'09 data sets. The TREC'07 and TREC'08 data sets include 45 and 50 assessed queries respectively and use Blog06 collection. The TREC'09 data set includes Blog08, a new collection of blogs, and has 50 new queries. We use only the title of the queries in our experiments.

The Blogs06 collection is a crawl of about one hundred thousand blogs over an 11-weeks period (Macdonald & Ounis, 2006a), and includes blog posts (permalinks), feeds and homepages for each blog. Blog08 is a collection of about 1 million blogs crawled over a year with the same structure as Blog06 collection. In our experiments we use only the permalinks component of the collections, which consist of approximately 3.2 million blog posts in the Blog06 and about 28.4 million blog posts in the Blog08 collection.

The Terrier Information Retrieval system[1] is used to index the collections and retrieve documents. For each query we select the top 15000 posts by

[1]http://ir.dcs.gla.ac.uk/terrier/

using the Terrier version of BM25 applying default stemming and stop words removal. Then we use the aggregation methods presented previously, to combine the relevance scores of posts for each blog. Since we have three years worth of data, we considered each year as a separate test collection. For setting the parameters, we learnt the parameter values for each year using the other two years as training data and tuned the parameters to optimize MAP.

Before generating the transition matrix for the random walk-based smoothing, we discarded terms with very high document frequency (more than 80% of the documents) or very low document frequency (less than 5 documents) in order to reduce the size of the graph. The length of the random walk is set to be long enough to approach the stationary distribution, our experiment showed that length of 20 is long enough for all the generated graphs.

We use the linguistic quantifier, as described in section 3.1, for calculating the weighting vector of OWA operators. Table 4 shows the properties of implemented OWA operators for different values of n. The a and b values for each quantifier are used in the formula 4 to calculate the weighting vector with the desired properties.

Tables 2, 3 and 4 show the evaluation results for TREC07, TREC08 and TREC09 respectively. Tables 1(a), 2(a) and 3(a) show the MAP (Mean Average Precision) and Precision at 10 (precision of the system in the top 10 retrieved documents) for the baseline, ExpCombSum and ExpCombMNZ, respectively for each year.

MAP values of the three different OWA operators with four different values of n on TREC07, TREC08 and TREC09 are shown in the Tables 1(b), 2(b) and 3(b) respectively. In order to test for statistical significance, we use the Wilcoxon signed-rank test on scores for each query at the 0.05 level. Statistically significant improvements over ExpCombSum and ExpCombMNZ, as the two baseline methods, are shown by † and ‡ respectively. It is shown that in most of the cases, OWA operators have statistically significant improvements over the baselines. As can be seen, aggregating the top 10 relevant posts in each blog has the best MAP across the three different data sets. In the best run for each year, the OWA aggregation operators improve the ExpCombSum in MAP by 35%, 33% and 31% for TREC07, TREC08 and TREC09 respectively.

It is worth to note that by increasing the number of aggregated posts, the operators with higher orness value perform better. We can see for most of the cases of aggregating top 5 posts, the *"As many as possible"* operator with the lowest orness value works the best. But for the higher number of aggregated posts like 20 or 30, *"At least half"* operator has always the best MAP value. It is mainly because of the low number of retrieved posts for each blog. In our experiments, average number of retrieved posts for each blog to be aggregated is 12.7, 13.73 and 7.66 for TREC07, TREC08 and TREC09 respectively. So when we aggregate higher number of posts with low orness operators, we give more importance to the smaller elements which are zero. Because of that, we miss some information that decreases the performance

of the system.

Precision at 10 for the discussed operators are shown in the Tables 1(d), 2(d), 3(d). We can see almost the same pattern for this evaluation metric as MAP. Same as noticed before, aggregating higher number of posts performs better when is done by lower orness operators and vice-versa. Statistically significant improvements over the baselines are detectable in most of the cases. In the best result for each year, OWA operators improve the $ExpCombSum$ by 24%, 14% and 38% for TREC07, TREC08 and TREC09 respectively.

Importance weighted OWA operators are implemented as described in section 3.2 by importance calculated in a graph representation that is explained in section 4. We exploit the same linguistic quantifier as before for defining scope function Q. Tables 1(c), 2(c) and 3(c) show MAP values of the OWA operators with cluster-based importance integrated in the weighting vector. It is shown that importance weighting improves previous OWA operators in most of the cases, specially for higher values of n. For the small values of n, since we have limited amount of information, small change in one of them can have a huge effect on the aggregated results and we can see that with $n = 5$, we do not get any improvement. On the other hand, with the higher values of n, we have more information and if the number of retrieved posts for a blog is less than n, the importance values for extra elements will be zero and they do not affect the result of the aggregation. Tables 1(e), 2(e) and 3(e) show the precision at 10 after involving importance in the OWA

operators.

6 Conclusion

The blog distillation task aims in retrieving blogs, as collections of posts, in response to a user information need. In this paper we used Ordered Weighted Averaging (OWA) operators with linguistic quantifiers for this task. We see each post as evidence about the relevance of the blog to the topic and rank the blogs based on the aggregation of their evidence. We show that, by using OWA with fixed number of highly relevant posts in each blog we can get statistically significant improvement over the baselines.

We carried out our experiments on TREC'07, TREC'08 and TREC'09 data sets. In the best runs by OWA operators, we got 35%, 33% and 31% improvements in MAP over ExpCombSum method for three consequent data sets respectively. These improvements over ExpCombSum in precision at 10 were 24%, 14% and 38% in the best results for each year.

We also investigate the extended version of OWA operators to integrate the importance of the elements in the aggregation. We calculate the importance of each post by its similarity with other highly relevant posts in a query-based cluster of documents. It was shown that this extension improves the performance of the retrieval, when the number of aggregated posts from each blog is high.

We have not modelled temporal properties of the posts and link structure

(a) MAP and Precision at 10 for baseline methods

Model	MAP	Precision at 10
ExpCombSum	0.2303	0.3778
ExpCombMNZ	0.1846	0.3333

(b) MAP of OWA operators with different quantifiers

	n=5	n=10	n=20	n=30
a=0.0 , b=0.5	0.2889 † ‡	**0.3121** † ‡	0.2979 † ‡	0.2675 † ‡
a=0.3 , b=0.8	0.3050 † ‡	0.2969 † ‡	0.2441 ‡	0.2083 ‡
a=0.5 , b=1.0	0.3096 † ‡	0.2800 † ‡	0.2218 ‡	0.1778

(c) MAP of importance weighted OWA operators

	n=5	n=10	n=20	n=30
a=0.0 , b=0.5	0.2486 ‡	0.2894 † ‡	**0.3090** † ‡	0.3011 † ‡
a=0.3 , b=0.8	0.2845 † ‡	0.3036 † ‡	0.2754 † ‡	0.2422 ‡
a=0.5 , b=1.0	0.3054 † ‡	0.2882 † ‡	0.2396 ‡	0.2011

(d) Precision at 10 of OWA operators with different quantifiers

	n=5	n=10	n=20	n=30
a=0.0 , b=0.5	0.4333 † ‡	0.4644 † ‡	**0.4711** † ‡	0.4511 † ‡
a=0.3 , b=0.8	0.4556 † ‡	0.4600 † ‡	0.4378 † ‡	0.4156 ‡
a=0.5 , b=1.0	0.4578 † ‡	0.4600 † ‡	0.4244 ‡	0.3867

(e) Precision at 10 of importance weighted OWA operators

	n=5	n=10	n=20	n=30
a=0.0 , b=0.5	0.3978 ‡	0.4578 † ‡	0.4644 † ‡	**0.4644** † ‡
a=0.3 , b=0.8	0.4444 † ‡	0.4556 † ‡	0.4511 † ‡	0.4444 † ‡
a=0.5 , b=1.0	0.4489 † ‡	0.4578 † ‡	0.4378 † ‡	0.4089 ‡

Table 2: Evaluation results for implemented methods over TREC07 data set. Statistically significant improvements over ExpCombSUM and ExpCombMNZ are indicated by † and ‡ respectively.

(a) MAP and Precision at 10 for baseline methods

Model	MAP	Precision at 10
ExpCombSum	0.1808	0.3300
ExpCombMNZ	0.1402	0.2580

(b) MAP of OWA operators with different quantifiers

	n=5	n=10	n=20	n=30
a=0.0 , b=0.5	0.2371 † ‡	**0.2422** † ‡	0.2052 † ‡	0.1775 ‡
a=0.3 , b=0.8	0.2371 † ‡	0.2073 † ‡	0.1475	0.1268
a=0.5 , b=1.0	0.2269 † ‡	0.1802 ‡	0.1306	0.1135

(c) MAP of importance weighted OWA operators

	n=5	n=10	n=20	n=30
a=0.0 , b=0.5	0.2155 ‡	**0.2347** † ‡	0.2317 † ‡	0.2167 † ‡
a=0.3 , b=0.8	0.2295 † ‡	0.2226 † ‡	0.1796 ‡	0.1510
a=0.5 , b=1.0	0.2307 † ‡	0.1953 ‡	0.1461	0.1285

(d) Precision at 10 of OWA operators with different quantifiers

	n=5	n=10	n=20	n=30
a=0.0 , b=0.5	0.3620 ‡	**0.3780** ‡	0.3420 ‡	0.3180 ‡
a=0.3 , b=0.8	0.3680 ‡	0.3340 ‡	0.3000 ‡	0.2740
a=0.5 , b=1.0	0.3620 ‡	0.3200 ‡	0.2860 ‡	0.2520

(e) Precision at 10 of importance weighted OWA operators

	n=5	n=10	n=20	n=30
a=0.0 , b=0.5	0.3380 ‡	0.3700 ‡	**0.3760** ‡	0.3580 ‡
a=0.3 , b=0.8	0.3600 ‡	0.3700 ‡	0.3120 ‡	0.3100 ‡
a=0.5 , b=1.0	0.3680 ‡	0.3360 ‡	0.3040 ‡	0.2760

Table 3: Evaluation results for implemented methods over TREC08 data set. Statistically significant improvements over ExpCombSUM and ExpCombMNZ are indicated by † and ‡ respectively.

(a) MAP and Precision at 10 for baseline methods

Model	MAP	Precision at 10
ExpCombSum	0.1601	0.2487
ExpCombMNZ	0.1306	0.2026

(b) MAP of OWA operators with different quantifiers

	n=5	n=10	n=20	n=30
a=0.0 , b=0.5	0.1887 † ‡	0.2047 † ‡	0.2090 † ‡	0.1995 † ‡
a=0.3 , b=0.8	0.2022 † ‡	**0.2099** † ‡	0.1873 † ‡	0.1683 ‡
a=0.5 , b=1.0	0.2061 † ‡	0.2036 † ‡	0.1712 ‡	0.1612 ‡

(c) MAP of importance weighted OWA operators

	n=5	n=10	n=20	n=30
a=0.0 , b=0.5	0.1680 ‡	0.1915 † ‡	0.2076 † ‡	**0.2116** † ‡
a=0.3 , b=0.8	0.1924 † ‡	0.2065 † ‡	0.2009 † ‡	0.1884 † ‡
a=0.5 , b=1.0	0.2039 † ‡	0.2080 † ‡	0.1841 † ‡	0.1694 ‡

(d) Precision at 10 of OWA operators with different quantifiers

	n=5	n=10	n=20	n=30
a=0.0 , b=0.5	0.2974 † ‡	0.3256 † ‡	**0.3436** † ‡	0.3410 † ‡
a=0.3 , b=0.8	0.3231 † ‡	0.3308 † ‡	0.3231 † ‡	0.3051 ‡
a=0.5 , b=1.0	0.3256 † ‡	0.3359 † ‡	0.3077 † ‡	0.2897 ‡

(e) Precision at 10 of importance weighted OWA operators

	n=5	n=10	n=20	n=30
a=0.0 , b=0.5	0.2974 ‡	0.3128 † ‡	0.3256 † ‡	**0.3436** † ‡
a=0.3 , b=0.8	0.3154 † ‡	0.3205 † ‡	0.3359 † ‡	0.3256 † ‡
a=0.5 , b=1.0	0.3205 † ‡	0.3256 † ‡	0.3128 † ‡	0.2923 ‡

Table 4: Evaluation results for implemented methods over TREC09 data set. Statistically significant improvements over ExpCombSUM and ExpCombMNZ are indicated by † and ‡ respectively.

of blogosphere here. In the future we intend to use this information to obtain a better relevance score or importance value for each post, before combining them.

Acknowledgment

We thank Mark Carman for his helps and constructive discussions. We also want to thank Amirhossein Malekpour for his great review and comments. This work was supported by Swiss National Science Foundation (SNSF) as XMI project (Project Nr. 200021-117994/1).

7 References

Arguello, J., Elsas, J., Callan, J., & Carbonell, J. (2008). Document representation and query expansion models for blog recommendation. *Proc. of the 2nd Intl. Conf. on Weblogs and Social Media (ICWSM)*.

Chiclana, F., Herrera-Viedma, E., Herrera, F., & Alonso, S. (2007). Some induced ordered weighted averaging operators and their use for solving group decision-making problems based on fuzzy preference relations. *European Journal of Operational Research*, *182*(1), 383–399.

Craswell, N., & Szummer, M. (2007). Random walks on the click graph. *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR confer-*

ence on Research and development in information retrieval (pp. 239–246). ACM.

Efron, M., Turnbull, D., & Ovalle, C. (2007). University of Texas School of Information at TREC 2007. *Proc. of the 2007 Text Retrieval Conf.*

Elsas, J. L., Arguello, J., Callan, J., & Carbonell, J. G. (2008). Retrieval and feedback models for blog feed search. *SIGIR* (pp. 347–354).

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

Hawking, D., & Thomas, P. (2005). Server selection methods in hybrid portal search. *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 75–82).

Kleinberg, J. M. (1999). Authoritative sources in a hyperlinked environment. *Journal of ACM*, *46*(5), 604–632.

Lee, Y., Na, S.-H., Kim, J., Nam, S.-H., Jung, H.-Y., & Lee, J.-H. (2008). Kle at trec 2008 blog track: Blog post and feed retrieval. *TREC*.

Macdonald, C., & Ounis, I. (2006a). The TREC Blogs06 collection: Creating and analysing a blog test collection. *Department of Computer Science, University of Glasgow Tech Report TR-2006-224*.

Macdonald, C., & Ounis, I. (2006b). Voting for candidates: adapting data fusion techniques for an expert search task. *Proceedings of the 15th ACM*

international conference on Information and knowledge management (pp. 387–396). ACM Press New York, NY, USA.

Macdonald, C., & Ounis, I. (2008). Key blog distillation: ranking aggregates. *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management* (pp. 1043–1052).

Macdonald, C., Ounis, I., & Soboroff, I. (2007). Overview of the trec-2007 blog track. *Proceedings of the Sixteenth Text REtrieval Conference*.

Nunes, S., Ribeiro, C., & David, G. (2008). Feup at trec 2008 blog track: Using temporal evidence for ranking and feed distillation. *TREC*.

Ounis, I., De Rijke, M., Macdonald, C., Mishne, G., & Soboroff, I. (2006). Overview of the TREC-2006 blog track. *Proceedings of TREC* (pp. 15–27).

Seo, J., & Croft, W. B. (2008). Blog site search using resource selection. *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management* (pp. 1053–1062). New York, NY, USA: ACM.

Soboroff, I., de Vries, A., & Craswell, N. (2006). Overview of the TREC 2006 Enterprise Track. *TREC 2006 Working Notes*.

Yager, R. (2009). Prioritized OWA aggregation. *Fuzzy Optimization and Decision Making*, *8*(3), 245–262.

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

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