Top Stories Identification From Blog to News In TREC 2010 Blog Track

Yu-Fan Lin, Jing-Hau Wang, Liang-Cheng Lai, Hung-Yu Kao
Department of Computer Science and Information Engineering
National Cheng Kung University
uf.lins@hotmail.com, westlife138@gmail.com, bens951@hotmail.com, hykao@mail.ncku.edu.tw

Abstract

In 2010 Blog Track, there are two tasks including Faceted Blog Distillation Task and Top Stories Identification Task. We mainly focus on the Top Stories Identification Task. In this task, there are two issues to solve. The first issue is ranking the important news stories on the specified day, named Story Ranking Task. The second issue is named News Blog Post Ranking Task. News Blog Post Ranking Task is ranking the blog posts that are relevant to the news story and diversifying the topics of blog posts.

In Story Ranking Task, our team Ikm100 (NCKU_CSIE_IKMLAB) submitted three runs. In the first run, a news story is scored by its number of discussion posts. In the second run, our idea is that if the news story is discussed by more people and the supporting blog post is relatively important, the news story would be more important. In the last run, we use the "Relevant-Post Time-Entropy evaluation" to score the news story.

In News Blog Post Ranking Task, we use the cosine similarity between the news story and the blog post, and also use importance of posts to extract the supporting blog posts of the news query.

1. Introduction

In TREC 2010, Blog track contains two tasks: Faceted Blog Distillation Task and Top Story Identification Task. Our team participates in the Top Story Identification Task. In TREC 2009, the Top Story Identification Task was a pilot search task addressing the problem of using blog data to identify top news stories [2].

In TREC 2010, the Top Story Identification Task has two stages [1]:

1. Story Ranking Task.
2. News Blog Post Ranking Task.

For the Story Ranking Task, it is different from the previous tasks in TREC2009. This task is treated as real-time event detection. We have the limitation of using information in Blog'08 data. With a given query date Q, all the information of Blog'08 we used must have the timestamp smaller than Q or equal Q. This limitation fit to mimic a real-time environment. Besides, for each query date we have to submit ranking of 100 news stories with 5 categories. The categories are "Business", "U.S.", "Sport", "SciTech" and "World".

For News Blog Post Ranking Task, the goal is to identify the top 50 relevant blog posts for each given news story with different period of time. Each ranking of blog posts should be diverse. It means that the blog posts should cover multiple aspects of the news stories. For each ranking, there are three different period of time described as follows:

1. The timestamp of blog posts should equal or smaller than the query timestamp
2. The timestamp of blog posts should equal or smaller than the query timestamp + 1 days
3. The timestamp of blog posts should equal or smaller than the query timestamp + 7 days

In this paper we first describe the data preprocessing in Section 2, then we introduce our method for the news stories ranking and the post ranking in Section 3 and Section 4, respectively. We report our performance in Section 5. Finally, the conclusion for our participation is in Section 6.

2. Data Preprocessing

The dataset in the blog track provided by TREC is the Blogs08 collection. It has various blogs from different blogospheres. We can extract a lot of information from this dataset, like timestamp, post titles, post contents, and post comments, etc. The preprocessing is a laborious work to exactly extract the information we need from every different blogs.

Blogs08 dataset contains feed files, permalink files, and blog homepages, and it is crawled over a 13-month period from early 2008 to early 2009. The crawled results contain 1,303,520 feeds and 28,488,766 permalink documents. In our experiment, we only focus on all feed files because they contain main sentences of each blog post and we think it is enough for this task. For each blog post, we extract the corresponding information from feed files, including title, content and so on.

In the second preprocessing step we filter out all non-English posts and stop words and apply the stemming process. Then we index each blog post title and content.

In this year, the data of news story is the TRC2 newswire corpus. The TRC2 is a collection of 1,613,707 news stories from Thomson-Reuters. First, we filter out the news which contains the error messages like “SERVICE ALERT”. Second, we remove the news stories in the document, “topNews-blacklist.docnos.txt.gz”. Then we create a separated index for each news story.

3. Approach for Story Ranking Task

Our three ranking methods are all based on the headline-post similarity network we build. We submitted three runs with run tags **Run1 (ikm100jing), Run2 (ikm100bindog),** and **Run3 (ikm100ufan)**. In Run1, we use the Sum of Cosine Similarity Approach. In Run2, we use the Average TF-IDF Approach. And in Run3, we use the Relevant-Post Time-Entropy Evaluation Approach. After ranking by our approaches, we do duplicated detection and block list for each run tag, and then get a set of ranking results as shown in Figure 1. The first set of ranking results, NReR, will be re-rank into a new set of ranking results as named ReR. Finally, NReR and ReR should be fitted to format of the output in the last step.

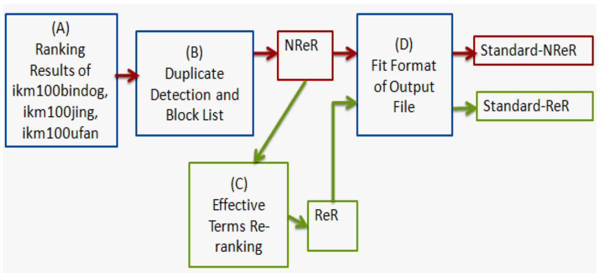

(A) Ranking Results of ikm100bindog, ikm100jing, ikm100ufan
(B) Duplicate Detection and Block List
NReR
(D) Fit Format of Output File
Standard-NReR
Standard-ReR
(C) Effective Terms Re-ranking
ReR

Figure 1 : Flow chart of NReR and ReR

3.1 Headline-Post Network

For each news corpus in TRC2, first, we build the weight of each word by TF-IDF (Term Frequency-Inverse Document Frequency) technique. There are two types of TF-IDF of a word in our methods. One is to use the information of headlines to build the weight of each word and is called TF-IDF$_h$. Another is to use the information of contents and is called TF-IDF$_c$.

Furthermore, we consider that the information of headlines is more important than the contents. Thus, we give each word a new value of TF-IDF weight as shown in the following equation:

$$\text{TF-IDF}_{\text{new}} = 2 * \text{TF-IDF}_{\text{h}} + \text{TF-IDF}_{\text{c}}$$

We also do the same TF-IDF calculating for the posts in Blogs08.

Second, for each “query date”, we calculate pair-wise cosine similarity between each news story and each blog post. For each query, the constructed network contains more than 100 million headline-post links. In order to filter out noises information in them, we only store the headline-post links which the similarity value is larger than 0.1.

3.1.1 Run1: Sum of Cosine Similarity Approach

$$\text{Score}(\text{h}) = \sum_{\text{p}\in\text{T}} \text{sim}(\text{h}, \text{p})$$

The score of each news story h is assigned as the sum of the cosine-similarity between news story h and post p that is posted during the timestamp T. In this run, we only consider the post whose cosine-similarity is larger than 0.3. The timestamp T is between the headline day d and d-1.

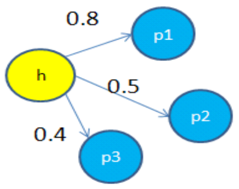


Figure 2 : Example for approach in Run1

For example, there are a news story h and three posts p1, p2 and p3. In Figure 2, the value of sim(h, p1) is 0.8 , sim(h, p2) is 0.5 , and sim(h, p3) is 0.4 . Then the news story gets the Score(h) that value is 1.7 .

3.1.2 Run2: Average TF-IDF Approach

The main idea in this run is that the news story is more important if it is discussed by more people, and a more important post has more important words. Here, we only use the posts on the query day T and only consider the post whose cosine-similarity is larger than 0.35. In next step, we calculate the average TF-IDF (post_avg_tfidf) for each blog post. The score is defined as:

$$\mathrm{Score(h)} = \sum_{p \in T} \mathrm{post\_avg\_tfidf(p)} * \mathrm{sim(h, p)}$$

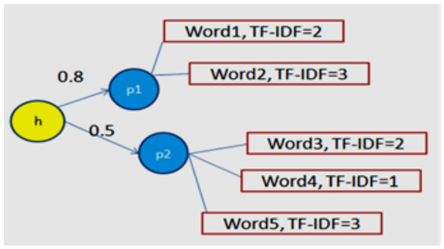


Figure 3 : Example for approach in Run2

Consider the example in Figure 3. The news story h has two associated posts p1 and p2. The value of sim(h, p1) is 0.8 and sim(h, p2) is 0.5. The post p1 contains Word1 and Word2. The post p2 contains Word3, Word4 and Word5. The score of h is then calculated as follows:

$$\mathrm{Score(h)} = \left(\frac{2+3}{2}\right) * (0.8) + \left(\frac{2+1+3}{3}\right) * (0.5)$$

3.1.3 Run3: Relevant-Post Time-Entropy Evaluation Approach

In Run3, we propose a method that collects and analyzes the entropy value of the posts, called "Relevant-Post Time-Entropy Evaluation". For a query date T, we use all the posts in the range from T-5 to T. In this run, we also consider the post whose cosine-similarity is larger than 0.35 to cut off the irrelevant posts.

In Figure 4, after extracting the publish time from dataset, the relevant posts are shown according to their published day on the time line. It means for the news story we can selected a set of relevant posts, and separate each posts by published day on the time line. We assume that before the hot story happened, this story may get higher attention and some bloggers would start to discuss the hot story in their posts. We call this case the posting-bursty behavior, as the example at 8/09 in Figure 4. We used the entropy value E to model the behavior as follows.

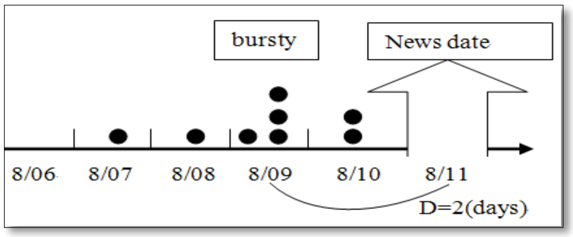


Figure 4 : Example of Relevant-Post Time-Entropy Evaluation

$$\mathrm{E} = -\sum_{i=T-5}^{T} q_i * \log(q_i)$$

'E' means the entropy value of relevant posts. 'q_i' means the probability of relevant posts appearing in date i. Each news story h has a score with the ratio between the entropy value and the bursty distance D, defined as

$$\mathrm{Score(h)} = \frac{(1-E)}{D},$$

where 'D' means the distance between the bursty date and the news date. Consider the example in Figure 4. Each black dot is a post. We find all the relevant posts with the news story which the date was from 8/06 to 8/11, and we find a bursty date 8/09. Then we get a distance D=2 between bursty date and news date in this example.

Then we calculate 'E' as follows:

$$\mathrm{E} = -\left\{0 + \left(\tfrac{2}{8}\right) * \left(\log\tfrac{2}{8}\right) + \left(\tfrac{4}{8}\right) * \left(\log\tfrac{4}{8}\right) + \left(\tfrac{1}{8}\right) * \left(\log\tfrac{1}{8}\right) + \left(\tfrac{1}{8}\right) * \left(\log\tfrac{1}{8}\right) + 0\right\} = 0.53,$$

and the score of h is calculated as follows:

$$\mathrm{Score(h)} = \frac{1 - 0.53}{2}$$

3.2 Duplicated Detection and Block List

In the TRC2 corpus, we found that there are many duplicated news in one subject. That is, if a news article is partially updated, it becomes a new version of the original news. Our goal is to detect the duplicated news after any revision. Our approach is described as follows:

1. We calculate the pair-wise similarity for each news story in query's ranking results.

2. For each news story h we extract the news story h' which has the higher similarity larger than 0.9.

3. For each h we have a news story group that contains h and h'. Then we save the latest news from the news story group and add the other news stories into the block list B.

Repeating the third step, we have the block list B for all news stories. Finally, we remove the news stories in the block list B for each query's news story ranking results.

Then we got a set of three ranking results of news stories with different runs: Run1, Run2, Run3, and we call this set of ranking results as Non Re-ranking Results (**NReR**). **NReR** will be re-ranked by the effective terms described in Section 3.3, and we will have a new set of ranking results, called Re-ranking Results (**ReR**).

3.3 Effective Terms Re-ranking

In this section, we are curious about whether some effective terms in the past hot news will have a positive influence for news stories ranking in the future or not. We build monthly effective terms list M_i for some month ID i, where i is the integer from 1 to 13. For example, M_3 present the effective terms list for March, 2008 and M_{13} specially presents for January, 2009.

First, we do not consider the news stories which do not have any word that has its IDF value larger than 1.5. We then select top 40 news stories in each run of **NReR** for each query.

Second, in order to build M_i, we select effective terms from the top 40 news stories of the queries in its previous month. For instance, the effective terms in M_5 is selected from the top 40 news stories from the queries in April, 2008. The queries in April from the dataset are 2008-4-2, 2008-4-19 and 2008-4-23. Besides, in order to build M_1, we add two queries of 2008-1-1 and 2008-1-2. We then choose effective terms from extracted news. Each term is selected when we thought it can be an important term in news stories. Each member is assigned 57 news of each month in average.

Third, for each query in i-th month, the news stories will be rescoring by the equation in the following equation.

$$\mathrm{Score}'(\mathrm{h}) = \prod_{\mathrm{w} \in \mathrm{M_i}} \mathrm{Score}(\mathrm{h}) * e^{\text{TF-IDF}(\mathrm{w},\mathrm{h})}$$

Score(h) means the original score of the news story h. Score'(h) means h is enhanced by those effective terms by the formula illustrated for each w belongs to M_i. The TF-IDF(w, h) means the TF-IDF value of the term w in the TRC2 news story h. If the w does not exist in h, then the value of TF-IDF(w, h) will be zero.

After the rescoring, we can re-rank the news stories and got a new set of re-ranking results (**ReR**). The re-ranking flow chart is also shown in Figure 1.

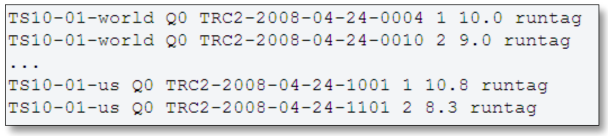
3.4 News stories Classification

Blog task of this year is different from tasks in last year. The difference is that we have to judge the category of each extracted news story. For a given time query, we have to return the top 100 news stories ranking for each category. Unfortunately, the TRC2 dataset do not contains category information of news. Thus, we use the tool, "Libsvm", to build a classifier to classify our news.

First, we crawled the news stories during the period from 2008/04 to 2008/06 for each 5 category as training set. There have total 1037 news stories for each category. Then we construct the inverted index of the training set, and we use the vector of TF-IDF values to represent each news story. We train a model according to the training set, and then use this model to predict the category of each news story in TRC2 corpus.

3.5 Format of Output File for Story Ranking Task

At the last step (D) shown in Figure 1, we have to fit the format of output files. For each numbered "query date", we select top 100 news stories ranking for each 5 categories. We already have the category of each news story from Section 3.4. Format of output file is like the sample [1] in Figure 5.



```
TS10-01-world Q0 TRC2-2008-04-24-0004 1 10.0 runtag
TS10-01-world Q0 TRC2-2008-04-24-0010 2 9.0 runtag
...
TS10-01-us Q0 TRC2-2008-04-24-1001 1 10.8 runtag
TS10-01-us Q0 TRC2-2008-04-24-1101 2 8.3 runtag
```

Figure 5 : Format of output file

Finally, we have two set of final results, Standard-NReR and Standard-ReR. Each set of final results contains 3 runs for our different approaches.

- Standard-NReR : Runs of Run1, Run2, Run3
- Standard-ReR : Runs of Run1, Run2, Run3

Our team submitted 3 runs for Story Ranking Task. We submitted 3 runs in Standard-ReR. And we will have comparison of statMap between Standard-NReR and Standard-ReR in Section 5.

Table 1 : Performance of submitted runs for Story Ranking Task

Group	Runs	TRC2-Fields	Mean StatMAP	StatMAP by Category				
				Business	**Sci-Tech**	**Sport**	**U.S.**	**World**
POSTECH_KLE	KLERUN1	HC	0.2206	0.1851	0.1821	0.1916	0.2458	0.2986
ICTNET	ICTNETTSRun2	HC	0.2138	0.0969	0.1898	0.2405	0.3025	0.2396
ikm100	Run1	HC	0.2151	0.1141	0.2483	0.1725	0.3897	0.1504
	Run2	HC	0.2107	0.1146	0.2390	0.1451	0.3870	0.1715
	Run3	HC	0.2043	0.0823	0.2425	0.1699	0.3827	0.1441

Table 2 : Performance of submitted runs for News Blog Post Ranking

Runs	Alpha-nDCG@10	P-IA@10	nERR-IA@10
Run1, α = 0.25	0.3335	0.1049	0.2907
Run2, α = 0.5	0.3750	0.1211	0.3332
Run3, α = 0.75	0.4075	0.1309	0.3720

Table 3 : The comparison of statMAP between Standard-NReR and Standard-ReR for Story Ranking Task

	Runs	Mean StatMAP	StatMAP of each category				
			Business	**Sci-Tech**	**Sport**	**U.S.**	**World**
Standard-NReR	Run1	**0.2339**	0.1200	0.2809	0.1716	0.4662	0.1307
	Run2	**0.2243**	0.1191	0.2725	0.1364	0.4393	0.1543
	Run3	**0.2139**	0.0874	0.2732	0.1704	0.4192	0.1201
Standard-ReR	Run1	0.2151	0.1141	0.2483	0.1725	0.3897	0.1504
	Run2	0.2107	0.1146	0.2390	0.1451	0.3870	0.1715
	Run3	0.2043	0.0823	0.2425	0.1699	0.3827	0.1441

4. Approach for News Blog Post Ranking Task

In this stage, we use the cosine similarity value between the headline and the post, and also use the count of post's comment to score the posts.

$$\text{Score}(p) = (\alpha)\big(\text{sim}(h, p)\big) + (1 - \alpha)(\text{Norm\_Comment})$$

sim(h, p) means the value of cosine similarity between the news story h and the blog's post p. "Norm_Comment" means the normalized value of count of post's comment. Term α means the weight of the cosine similarity comparing with count of post's comment.

For each news story of query, we selected relevant posts with their cosine similarity larger than 0.35. Before we score and rank the relevant post. We classify posts to 5 categories ("Business", "U.S.", "Sport", "SciTech" and "World"). After post ranking, we select the post which only has the same category with the news story to submit our results. For example, if the query of the news story belongs to business category, then we only rank those posts which belong to the business category. We submitted 3 runs with three different α in this stage. We set α = 0.25, 0.5, and 0.75 for each run respectively.

5. Results of Runs

Our team IKM100 submitted 3 runs named Run1 (ikm100jing), Run2 (ikm100bindog) and Run3 (ikm100ufan) in Stories Ranking Task. All runs are automatically generated and ranked. The performance of our runs and runs of other participants in Story Ranking Task is shown in Table 1.

In News Blog Post Ranking Task, our team submitted 3 runs named 'Run1', 'Run2' and 'Run3' with different α values shown in Table 2. We select the evaluation of alpha-nDCG@10, P-IA@10 and nERR-IA@10 to show our performance as shown in Table 2.

TREC organizers have provided test programs of evaluation on TREC website. We have download the program files of blog story ranking task and implement the evaluation process on Ubuntu OS. We also implement some extra evaluation. The comparison of statMAP between Standard-NReR and Standard-ReR is shown in Table 3.

6. Conclusions

In TREC 2010, we focus on the Story Ranking of the Top Story Identification Task. We propose three general methods based on the headline-post network. In this network, we identify each cosine similarity between post and news story. For the category classification, our team crawl the news from Reuters website and use the "Libsvm" to classify news stories in TRC2 dataset. Additionally, we found something interesting that the effective term list we retrieved is not as useful as we expected. The performance of Standard-NReR outperformed Standard-ReR according to statMAP. This means that the influence of term list will shift with time and a better approach should be developed to find the useful effective terms.

In the News Blog Post Ranking, our method does not use diversity features to rank the posts. We use the blogger's attention feature, i.e. the number of post's comments. This feature helps us to identify popular post for the news story. We also use cosine similarity to judge the relevance between post and news story.

In the future work, we focus on exploring links between blogs and applying other suitable models to the Top Story Identification Task. The work will try to quantify blogger's attention to the top stories and see how we can use more characteristics of the blogosphere.

7. Acknowledgement

We would like to appreciate the TRC2 newswire corpus from Thomson-Reuters, and the technological guidance of using evaluation program from the organizers in blog track.

8. References

1. *http://ir.dcs.gla.ac.uk/wiki/TREC-BLOG*. 2010.
2. Macdonald, C., et al., *Blog track research at TREC*. SIGIR Forum, 2010. **44**(1): p. 58-75.