FDU at TREC 2007: opinion retrieval of Blog Track

Qi Zhang, Bingqing Wang, Lide Wu, Xuanjing Huang

Abstract

This paper describes our participation in the opinion retrieval task at Blog Track 07. The system consisted of the preprocess part, the topic retrieval part and sentiment analysis part. In the topic retrieval part, we adopted pseudo-relevance feedback and a novel approach to form a modified query. In the sentiment analysis part, each blog post was given an opinion score based on the sentences contained in this post. The subjectivity of each sentence was predicted by a CME classifier. Then the blog posts were reranked based on the similarity given by the topic retrieval and the opinion score given by the sentiment analysis.

1 Introduction

The opinion retrieval task[1] was aimed to explore the information seeking behaviour in the blogosphere. The large scale test collection, the TREC Blog06 collection[2], was used again in Blog Track 07.

Our approach to this task was a three-step process. A preprocess was first conducted to extract the content from the permalink HTML pages. Lucene[1] was used to build the index on the preprocessed corpora. Then in the topic retrieval part, we retrieved the top 2000 blog posts for each topic. The expansion terms for a query were extracted by pseudo-relevance feedback. A machine learning approach was developed to select the expansion terms aiming at raising the MAP. In the sentiment analysis part, a CME classifier was trained to predict the subjectivity of each sentence in a blog post. Then a SVM classifier gave an opinion score for this blog post based on the sentence-analysis. Finally, all the blog posts were reranked based on the similarity given by the topic retrieval and the opinion score given by the sentiment analysis. The top 1000 blog posts were submitted.

The remainder of this paper is structured as follows. Section 2 provides an overview of the system. Section 3 describes the topic retrieval part. Section 4 describes the sentiment analysis part. Section 5 introduces the submitted runs and the evaluation result. Conclusions are made in section 6.

[1]http://lucene.apache.org/

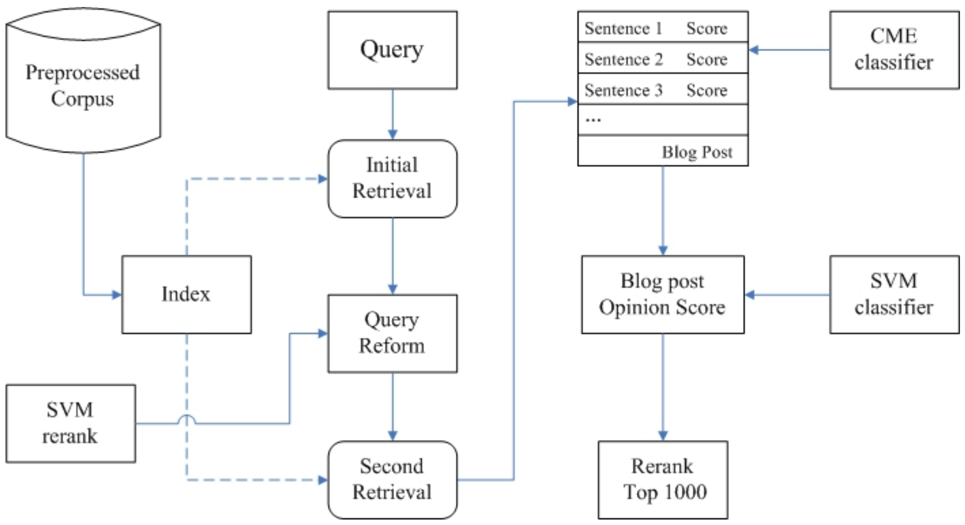

Figure 1: System Architecture

2 System Overview

The system was composed by the preprocess part, the topic retrieval part and the sentiment analysis part. The system architecture is shown in Figure 1.

The preprocess part was designed to extract the content of the permalink HTML pages, which would be discussed in section 2.1.

The topic retrieval part retrieved 2000 blog posts for each of the 50 topics. Pseudo-relevance feedback extracted expansion query terms from the initial retrieval result. The top t expansion terms together with the weight were generated from the top k returned documents. A SVM was trained to rerank these expansion terms.

The sentiment analysis gave each blog post an $OpScore$(opinion score). A CME classifier was trained to predict the subjectivity for each sentence in a blog post. This CME classifier was trained on the movie review data. Based on the sentence-level prediction, the $OpScore$ for this blog post was predicted by a SVM classifier.

Combining the $OpScore$ with the similarity by the topic-retrieval part, the 2000 blog posts were reranked and the top 1000 were selected.

2.1 Corpus and Preprocess

The TREC Blog06 collection contains the permalinks HTML pages, the feed file and the blog homepage. We only use the permalink HTML pages for the opinion retrieval task. These permalinks HTML pages were in different style and some of them were poorly structured, filled with noisy information.

For each blog post, we discarded the hyper-links, the script and the style information in the web page and filter all the html tags. Some html pages were damaged and we only filtered the html tags and hyper-links in those damaged-structure html pages.

Before the preprocess, the permalink HTML pages amounted to about 80G, after the preprocess the cleaned permalink HTML pages amounted to about 18G.

3 Topic Retrieval

Based on the preprocessed corpora, index was build using Lucene. Pseudo-relevance feedback modified the original query by adding expansion terms from the initial retrieval. Expansion terms were further selected using machine learning approach.

3.1 Pseudo-Relevance feedback

For the vector space model, query expansion approach had been discussed to reformulate the query so that it could be closer to the term-weight vector space of relevant documents. The standard Rochio formulation[4] generated a modified query $\overrightarrow{q_m}$ as follows

$$\overrightarrow{q_m} = \alpha \overrightarrow{q} + \frac{\beta}{|D_r|} \sum_{\forall \overrightarrow{d_j} \in D_r} \overrightarrow{d_j} - \frac{\gamma}{|D_n|} \sum_{\forall \overrightarrow{d_j} \in D_n} \overrightarrow{d_j}$$

D_r : set of relevant documents among the retrieved documents.
D_n : set of non-relevant documents among the retrieved documents.
α, β, γ are constants.

For the pseudo-relevance feedback in our case, it was assumed that the top k documents from the initial retrieval made up of the relevant document set D_r, the non-relevant document set D_n was hard to define and omitted here. Then the query was reformed as follows

$$\overrightarrow{q_m} = \overrightarrow{q} + \alpha \sum_{\forall \overrightarrow{d_j} \in D_r} \overrightarrow{d_j} = \overrightarrow{q} + \alpha \overrightarrow{d}$$

$\overrightarrow{q}$ was the original query, $\overrightarrow{d_j}$ represented the term weight vector of document j, $\overrightarrow{d_j} = (w_{1,j}, w_{2,j}, \cdots, w_{n,j})$. And

$$\overrightarrow{d} = \sum_{\forall \overrightarrow{d_j} \in D_r} \overrightarrow{d_j} = \sum_{\forall \overrightarrow{d_j} \in D_r} (w_{1,j}, w_{2,j}, \cdots, w_{n,j}) = (w_1, w_2, \cdots, w_n)$$

$w_{i,j}$ was the term weight of word i in document j, and $w_i = \sum w_{i,j}$ was the term weight of word i in the local collection D_r. We used a heauristic modified $w_{i,j} = tf_{i,j} * idf_i / sidf_i$, where $sidf_i$ was the idf value of word i in the local collection D_r, while the idf_i was the idf value of word i in the global collection. This formulation decreased the weight of those terms, which had similar distribution in the local document set D_r and the global collection.

The vector $\overrightarrow{d}$ contained the term weight of all the words. $\alpha = \frac{0.5}{max(w_i)}$ was a factor, which normalized every term weight in $\overrightarrow{d}$ and made any term weight lower than 0.5. We assumed that the weight of any expanded term should be less than the original query.

3.2 Expanded Terms Reranked by SVM

We needed the top t terms together with the weight to form a modified query. This could be done by ranking all the term weight in $\overrightarrow{d}$ to find the top t terms. However, we hoped to find out a machine learning approach for the expansion term selection. These two approaches were both tried in our submitted runs.

A SVM was trained to predict on the expanded terms. The advantage using a machine learning approach was that the expanded term selection could be conducted more reliable compared with ranking the terms by empirical expression. Our approach provided a method to generate the training data set, to tell whether one expanded term was more related to the topic than another expanded term.

The training set of instance-label pairs $(\boldsymbol{x_i}, y_i)$ were generated from Blog Track 06. $\boldsymbol{x_i} \in \mathbb{R}^n$ representing the feature vector of the expanded term i and y_i is a label indicating the closeness of this term with the topic. SVM^{light} was used to train a RBF regression module. The training data was generated as follows.

1. original query q was retrieved and evaluated by trec_eval, the MAP referred as $BaseMAP$
2. expanded terms of this query were extracted. $(t_1, t_2, \cdots)$
3. for the expanded term t_i, build the feature vector $\boldsymbol{x_i}$
4. q and t_i form a new query q_i, which was retrieved and evaluated by trec_eval, the MAP referred as $TermMAP_i$
5. $y_i = \frac{TermMAP_i}{BaseMAP}$, the pair $(\boldsymbol{x_i}, y_i)$ was an instance of the training data

The feature vector of the expanded term t_i, $\boldsymbol{x_i} \in \mathbb{R}^n$ was defined as

$$\boldsymbol{x_i} = (sumtf,\ avgTf,\ maxTf,\ idf,\ sdf,\ sidf,\ (idf - sidf),\ sumtfidf,\ sumtf\frac{idf}{sidf})$$

where for expanded term t_i, $sumtf$ was the sum of term frequency of every document in D_r, $avgTf$ was the average term frequency in D_r, $maxTf$ was the maximum term frequency in D_r, *idf* was the idf in the global collection, *sdf* was the document frequency in D_r, *sidf* was the idf in D_r.

The label $y_i = \frac{TermMAP_i}{BaseMAP}$ indicated the relationship of expanded term t_i with the topic. The closeness of the expanded term was calculated by how much this term could help to raise the MAP when adding it to the original query. $y_i > 1$ meant this term was useful in raising the performance, while $y_i < 1$ meant this term was less related.

SVM^{Light} was used to do the RBF regression. Blog Track 06 provided two kinds of assessment on the run, the opinion retrieval results and the topic-relevance results. Correspondingly, two kinds of regression module were trained on these two assessments. The module trained on the topic-relevance results was aimed to raise the topic-relevance MAP with no sentiment analysis feature. The module trained on the opinion retrieval results was aimed to raise the opinion retrieval MAP, which had sentiment analysis feature to some extense.

For a topic, empirically, the top *120* documents composed of the D_r, the top *400* expanded terms were predicted and reranked. The top *200* expanded terms were finally added to the original query to form a new one. The top 2000 blog posts were returnedby retrieving this new modified query. Each blog post was assigned a similarity score *Sim* representing the relevance of this post with the query.

4 Sentiment Analysis

Sentiment Analysis in opinion retrieval task focused on the classification between the opinion blog post and non-opinion blog post. First, each sentence of a blog post was given a sentence score indicating the subjectivity of this sentence. Then, based on all the sentences in this blog post, an *OpScore*(opinion score) was given to indicate the subjectivity of this blog post by SVM. Finally, the *OpScore* given by the sentiment analysis and *Sim*(similarity) given by the topic retrieval was combined to form a final score.

4.1 Sentence Evaluation and Blog post Evaluation

For the sentence evaluation, each sentence was predicted by a CME classifier. The training data for the CME classifier was from the movie review data[2]. The subjectivity data set v1.0 of movie-review data contains 5000 subjective and 5000 objective processed sentences. The features for the CME classifier

[2]http://www.cs.cornell.edu/People/pabo/movie-review-data/

were unigram, bigram, whether this sentence contains words in opinion. The opinion word was detected by General Inquirer Lexicon[3].

For the blog post evaluation, a SVM classifier was trained to predict the $OpScore$(opinion score) for a blog post based on all the sentences in this blog post. The training data $(\boldsymbol{x_i}, y_i)$ were generated from the Blog Track 06. The training data was opinioned relevant documents(labeled as $y_i = +1$) and non-opinioned but relevant documents(labeled as $y_i = -1$).

The feature $\boldsymbol{x_i}$ was generated from all the sentences in the $i - th$ blog post. A blog post was transformed into a vector. Each element in the vector corresponded to a sentence. Suppose there were l sentences in a blog post, each sentence got a sentence score describing the subjectivity or objectivity. The blog post could be represented as follows,

$$\overrightarrow{d_O} = (s_1, s_2, \cdots, s_l), \quad \begin{cases} s_i > 0 & \text{sentence } i \text{ is subjective} \\ s_i < 0 & \text{sentence } i \text{ is objective} \end{cases}$$

Then, another two vector describing the subjectivity and the objectivity could be derived from $\overrightarrow{d_O}$.

$$\overrightarrow{d_{Sub}} = (s_1, s_2, \cdots, s_l), \quad \begin{cases} s_i > 0 & \text{sentence } i \text{ is subjective} \\ s_i = 0 & \text{sentence } i \text{ is objective} \end{cases}$$

$$\overrightarrow{d_{Obj}} = (s_1, s_2, \cdots, s_l), \quad \begin{cases} s_i = 0 & \text{sentence } i \text{ is subjective} \\ s_i < 0 & \text{sentence } i \text{ is objective} \end{cases}$$

A sentence was taken as relevant sentence if this sentence contained the term in the original query. The document was presented as follows

$$\overrightarrow{d_T} = (t_1, t_2, \cdots, t_l), \quad t_i = \begin{cases} 1 & \text{if sentence contain query term} \\ 0 & \text{otherwise} \end{cases}$$

Then, a fuzzy relevant sentence vector could be derived from $\overrightarrow{d_T}$. If the previous two sentences contained the query term, this sentence could also be labeled as relevant sentence.

$$\overrightarrow{d_F} = (t_1, t_2, \cdots, t_l), \quad t_i = \begin{cases} 1 & \text{previous 2 sentences contain query term} \\ 0 & \text{otherwise} \end{cases}$$

Features $\boldsymbol{x_i}$ were generated from the vector d_{Sub}, d_{Obj} and d_T, d_F. The features we used are described as follows.

[3]http://www.wjh.harvard.edu/ inquirer/

Feature	**Description**
subscore_relbase	$d_{Sub}^T \cdot d_T$
subscore_relwin	$d_{Sub}^T \cdot d_F$
objscore_relbase	$d_{Obj}^T \cdot d_T$
objscore_relwin	$d_{Obj}^T \cdot d_F$
rel_base_cnt	# relevant sentences
rel_base_cnt	# fuzzy relevant sentences
relbase_sub_cnt	# sentences both relevant and subjective
relbase_obj_cnt	# sentences both fuzzy relevant and objective
relwin_sub_cnt	# sentences both relevant and objective
relwin_obj_cnt	# sentences both fuzzy relevant and objective
sim_subscore_relbase	$d_{Sub}^T \cdot d_T/(1+\|d_{Sub}\|)(1+\|d_T\|)$
sim_subscore_relwin	$d_{Sub}^T \cdot d_F/(1+\|d_{Sub}\|)(1+\|d_F\|)$
sim_objscore_relbase	$d_{Obj}^T \cdot d_T/(1+\|d_{Obj}\|)(1+\|d_T\|)$
sim_objscore_relwin	$d_{Obj}^T \cdot d_F/(1+\|d_{Obj}\|)(1+\|d_F\|)$

4.2 Similarity and Opinion Score

The opinion score($OpScore \in (-1, 1)$) given by the SVM classifier was integrated with the similarity($Sim \in (0, 1)$) given by the topic retrieval.

The opinion score was normalized to $(0, 1)$ by a logistic function, the logistic opinion score($LogOpScore$) was defined as

$$y = \frac{1}{1+e^{-\lambda x}}, \quad \begin{cases} \lambda = 5 & if\ x < 0 \\ \lambda = 3 & if\ x > 0 \end{cases}$$

The final score for a blog post was $Score = LogOpScore * Sim$, all the blog posts were reranked by the final score.

5 Submission and Evaluation Results

We submitted 6 automatic runs, as follows:

- FDUNoOpTisd: A baseline run only using the pseudo-relevance feedback without SVM regression module
- FDUNOpRSVMT: the FDUNoOpTisd with SVM regression module trained on topic-relevance results
- FDUTisdOpSVM: the FDUNoOpTisd with SVM regression module trained on opinion retrieval results, this module has some feature of sentiment analysis
- FDUNoOpTSem: the FDUNoOpTisd run with sentiment analysis

- FDUTNRSVMSem: the FDUNOpRSVMT run with sentiment analysis
- FDUTOSVMSem: the FDUTisdOpSVM run with sentiment analysis

Table 1: Opinion Finding Result

Run	MAP	R-prec	b-Pref	P@10
baseline	0.2388	0.3011	0.3083	0.3680
FDUNoOpTisd	0.2992	0.3351	0.3357	0.4340
FDUNOpRSVMT	0.3178	0.3447	0.3498	0.4520
FDUTisdOpSVM	**0.3179**	**0.3467**	**0.3501**	**0.4540**
FDUNoOpTSem	0.3019	0.3382	0.3381	0.4460
FDUTNRSVMSem	0.3141	0.3475	0.3496	0.4620
FDUTOSVMSem	0.3143	0.3465	0.3499	0.4600

Table 2: Topic Relevance Result

Run	MAP	R-prec	b-Pref	P@10
baseline	0.3927	0.4520	0.5222	0.6340
FDUNoOpTisd	0.4506	0.4744	0.5272	0.6320
FDUNOpRSVMT	0.4709	0.4888	0.5428	0.6520
FDUTisdOpSVM	**0.4714**	**0.4889**	**0.5432**	**0.6540**
FDUNoOpTSem	0.4355	0.4626	0.5113	0.6500
FDUTNRSVMSem	0.4484	0.4765	0.5228	0.6620
FDUTOSVMSem	0.4488	0.4768	0.5232	0.6620

The evaluation results of the 6 submitted runs are listed in the table 1 and table 2. All these submitted results were using query expansion techniques, so we added a “baseline” result for comparison. This “baseline” run didn’t use any query expansion or sentiment analysis module. Our best run “FDUTisdOpSVM” is emphasized in the table.

From the table above, we can find that the query expansion approach using SVM to rank the expanded terms could get better performance than using the empirical expansion.

6 Conclusion

We described our system in this paper. The opinion retrieval task required the combination of information retrieval techniques and the sentiment analysis approaches.

We developed a pseudo-relevance feedback approach by empirical query reformulation. A novel approach was developed to rerank the expanded terms by machine learning method. The sentiment analysis was based on sentence-level analysis. In future work, we intend to make a further exploration on the expansion approach using machine learning method and the sentiment classification of a blog post.

Acknowledgements

This work was partially supported by NSF of China under the grant of 60673038.

References

[1] Iadh Ounis, Maarten de Rijke, Craig Macdonald, Gilad Mishne, Ian Soboroff "Overview of the TREC-2006 Blog Track", *In TREC 2006*, 2006

[2] Craig Macdonald and Iadh Ounis. "The TREC Blog06 Collection : Creating and Analysing a Blog Test Collection" *DCS Technical Report TR-2006-224*. Department of Computing Science, University of Glasgow. 2006.

[3] Ricardo Baeza-Yates, Berthier Ribeiro-Neto "Modern Information Retrieval" pp. 117-118

[4] J. J. Rochio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System-Experiments in Automatic Document Processing.* Pretince Hall Inc., Englewood Cliffs, NJ, 1971.