Combining Language Model with Sentiment Analysis for Opinion Retrieval of Blog-Post

Xiangwen Liao, Donglin Cao, Songbo Tan, Yue Liu, Guodong Ding and Xueqi Cheng
Intelligent Software Department, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
tansongbo@software.ict.ac.cn, liuyue@ict.ac.cn

Abstract

This paper describes our participation in Blog Opinion Retrieval task this year. We conduct experiments on "FirteX" platform that is developed by our lab. Language Model is used to retrieve related blog unit. Interactive Knowledge is adopted to expand query for retrieve blog unit include opinion. Then we introduce a novel extracting technology to extract text from retrieved blog-post. Finally, Lexicon based method is used to rerank the document by opinion.

1 Introduction

Since opinion retrieval is a new retrieve task emerging this year, our aim to Blog Opinion Retrieval task is to construct an opinion retrieval framework using Blog data as an important test bed. Compared with traditional retrieval task, opinion retrieval is more challenged since it is very difficult to judge opinion about some topic.

After filtering spam permalink data, we obtain about 80G permalink data. Statistics Language Model [1] is used to retrieve related blog unit. Interactive Knowledge is adopted to expand query for retrieve blog unit including opinion. Then we introduce a novel extracting technology to extract text from the retrieved result. In order to rank the documents with stronger opinion in topper position, Lexicon [2] based method is used to rerank the documents.

2 Retrieve Model

In our system, we used the basic language modeling approach. The language modeling approach to IR is attractive and promising because it connects the problem of retrieval with that of language model estimation. The basic idea behind it can be described as follows.

Suppose we have a collection with total N documents, whose vocabulary is $V = \{w_1, w_2, \ldots, w_L\}$. Given a query $Q = q_1 q_2 \ldots q_m$ ($q_k \in V$, $1 \leqslant k \leqslant m$), the task is to rank the documents according to the relevance of each document with the query. Based on the assumption of words independence, the language model approach views each document D as an observed sequence of words generated by a multinomial language model parameterized by $\boldsymbol{\theta}_D = (\theta_{D1}, \theta_{D2}, \ldots, \theta_{DL}) \in [0,1]^L$, where $\theta_{D1} + \theta_{D2} + \ldots + \theta_{DL} = 1$ and $\theta_{Di} = p(w_i|\boldsymbol{\theta}_D)$, the probability of observing word w_i under the multinomial generation model (i.e. unigram model) parameterized by $\boldsymbol{\theta}_D$. And the relevance of document D with the query Q is measured by the likelihood of Q according to the multinomial model $\boldsymbol{\theta}_D$:

$$P(Q|\boldsymbol{\theta}_D) = \prod_{i=1}^{m} p(q_i|\boldsymbol{\theta}_D) = \prod_{w \in Q} p(w|\boldsymbol{\theta}_D)^{c(w,Q)} \quad (1)$$

where $c(w,Q)$ is the frequency of word w occurring in query Q.

As shown in equation (1), the retrieval problem is now essentially reduced to a multinomial language model estimation problem. The simplest method to estimate $\boldsymbol{\theta}_D$ is to utilize the maximum likelihood estimator (MLE):

$$p(w \mid \hat{\boldsymbol{\theta}}_D) = p_{ML}(w \mid \hat{\boldsymbol{\theta}}_D) = \frac{c(w,D)}{|D|} \tag{2}$$

where $c(w,D)$ is the frequency of word w occurring in document D and $|D|$ is the length of D, i.e. $|D| = \sum_w c(w,D)$.

However, using MLE can cause serious zero probability problem due to data sparseness. A direct solution to this problem is smoothing, which adjusts the MLE so as to assign a nonzero probability to the unseen words and improve the accuracy of language model estimation. According to [5], there are three popular smoothing methods applied to the language modeling approaches to ad hoc IR: Jelinek-Mercer, Dirichlet and Absolute Discounting, summarized in Table 1. Notice that $|D|_u$ is the number of unique words in document D. The collection language model $\boldsymbol{\theta}_C$ is typically estimated by MLE based on the whole collection, i.e.

$$p(w \mid \hat{\boldsymbol{\theta}}_C) = p_{ML}(w \mid \hat{\boldsymbol{\theta}}_C) = \frac{\sum_D c(w,D)}{\sum_w \sum_D c(w,D)} \tag{3}$$

Table 1. Summary of the three popular smoothing methods

Jelinek-Mercer	$p(w \mid \hat{\boldsymbol{\theta}}_D) = \lambda p_{ML}(w \mid \hat{\boldsymbol{\theta}}_D) + (1-\lambda) p(w \mid \hat{\boldsymbol{\theta}}_C)$
Dirichlet	$p(w \mid \hat{\boldsymbol{\theta}}_D) = \frac{c(w,D) + \mu p(w \mid \hat{\boldsymbol{\theta}}_C)}{\lvert D \rvert + \mu}$
Absolute Discounting	$p(w \mid \hat{\boldsymbol{\theta}}_D) = \frac{\max(c(w,D) - \delta, 0)}{\lvert D \rvert} + \frac{\delta \lvert D \rvert_u}{\lvert D \rvert} p(w \mid \hat{\boldsymbol{\theta}}_C)$

In our experiments, we utilized the Dirichlet smoothing method for the document language model estimation, where the smoothing parameter u is set to 2000.

3 Query Expansion Using Interactive Knowledge

Query expansion is an effect method to bridge the semantic gap between the vocabulary of documents and that of users. There are three kinds of traditional query expansion approaches: Lexicon based Query Expansion, Global-Analysis Based Query Expansion [6], Local-Analysis based Query Expansion [4]. Lexicon based Query Expansion uses lexicon, such as WordNet, to select terms semantically related to Query terms. Global-Analysis Based Query Expansion selects terms co-occurring in the Corpus with Query terms, while Local-Analysis based Query Expansion selects terms co-occurring in the top-N documents with Query terms.

Query expansion can be performed either manually or automatically. During opinion retrieval task, we are concerned with semi-automatic query expansion. For each query, traditional query expansion often selects expansion term by co-occurrence statistics. We first classify each query into different categories. For example, the query “March of the Penguins” belongs to film category. And then select terms that could describe the firm from Inquirerbasic lexicon [2]. After obtaining the first initiated result, we get the adj terms around the query from top-50 documents and then manually select some terms much more suitable to describe the film. Finally, we use these terms to expand the query and retrieve the top 1000 documents.

4 Extracting Post and Comment Based on Feature Statistics

Blog Extraction is a kind of difficult job because each blog site has a lot of different blog templates which can be chosen by blogers. Each template has different styles which greatly enhance the difficulty of extraction. Under this condition, we cannot easily build one extracting method for any blog template. In order to address this issue, we need to build up a common infrastructure which grasps the essential factors of blog pages. With the aim to achieve this goal, we combine the similarity of html tag tree with the similarity of content to find the most suitable extracting range.

Based on above idea, we propose a novel extraction algorithm that can effectively extract the main texts and comments from blog pages. It is described as Figure 1.

Extracting algorithm:

1. Locate the range of main text.
   - 1.1 Build a html tag tree.
   - 1.2 For each html tag, calculate the number of word in it and its position in vision.
   - 1.3 Choose a tag which has more words and its position is wider than other sibling tags in html tag tree.
   - 1.4 Use the range of chosen html tag as the main text range.
2. Locate the title of main text.
   - 2.1 For each line in range, find the lines which are highlighted or have bigger font size.
   - 2.2 Use the words in "<title>" as a clue to decide which selected line is title.
3. Locate the start of comments.
   - 3.1 Find those lines which have some key words of start of comment, like "comment", "reply".
   - 3.2 Choose the line which has more few words and has bigger font size.
4. Calculate the number of comments.
   - 4.1 From the start of comments, find each line which has time expression.
   - 4.2 Compute the difference of those selected lines and choose the more similar lines as a symbol of comment.
   - 4.3 Compute the start and end of each comment through the html tag tree.

Figure1: The outline of extracting algorithm

5 Lexicon based Opinion Ranking

The sentiment Lexicon we used for identifying positive and negative terms is taken from the General Inquirer [2] (GI for simplicity), which is a dictionary that contains information about English word senses.

GI is a system which lists terms as well as different senses for the terms. For each sense it provides a short definition as well as other information about the term. This includes tags that label the term as being positive, negative, a negation term, an overstatement, or an understatement. For example, there are two senses of the word "fun". One sense is a noun or adjective for enjoyment or enjoyable. The second sense is a verb that means to ridicule or tease, to make fun of. The first sense of the word is positive, marked as "Positiv", while the second is negative, marked as "Negativ". There are other labels for each sense.

One difficulty in using this lexicon is that English word always contains many inflections. For example, the word "take" may has four inflections: "takes", "taking", "took", "taken". In order to

address this problem we employed the Porter stemming algorithm2 [3]. This algorithm does not produce a lemma for a term, but rather maps similar words to a string. After removing word suffix, we obtain 1743 negative words and 1313 positive words.

The method we used to classify a blog-post is very simple and straightforward, that is, only to count positive and negative terms in it. If the blog-post contains more positive than negative terms we deem it as positive, and if the number of negative terms is greater than the number of positive terms we assign it as negative. If a blog-post contains an equal number of positive or negative terms, we say it is neutral.

6 References

[1] [Ponte98] Ponte J., W. Croft. (1998). A Language Modeling Approach to Information Retrieval. In Proceedings of SIGIR1998, 275-281.

[2] Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, Daniel M. Ogilvie, and associates. The General Inquirer: A Computer Approach to Content Analysis. The MIT Press, 1966.

[3] Martin F. Porter. An algorithm for suffix stripping. Program, 14(3): 130-137, 1980.

[4] Buckley, C., Salton, G., Alan, J., and Singhal, A. 1995. Automatic query expansion using SMART. In Proceedings of the 3rd Text Retrieval Conference (TREC-3), D. Harman, Ed. National Institute of Standards and Technology, Gaithersburg, MD, 69–80.

[5] MacKay, David J. C. and Linda C. Peto. A hierarchical Dirichlet language model. Natural Language Engineering, 1(3): 1-19, 1995.

[6] Sparck Jones, K. 1971. Automatic Keyword Classification for Information Retrieval. Butterworths, London.