IIT Kharagpur at TAC 2008: Statistical Model for Opinion Summarization

Sushant Kumar and Diptesh Chatterjee
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
India – 721302
sushant3d@gmail.com, dipteshc@iitkgp.ac.in

Abstract

In this paper we present our participation at TAC 2008 opinion summarization task. We make use of a statistical model for opinion extraction and the subsequent summarization of the extracted opinions. We also present its performance as per the official TAC 2008 evaluation results.

1 Introduction

The TAC 2008 Opinion Summarization task was to generate query-based summary of opinions as present in the given collection of blog documents. One of the largest sources of information on the web is in the form of weblogs, where people express their opinions on a variety of topics ranging from politics, movies, music to new products that have hit the market. So it is quite an obvious fact that if we can harvest the information contained in these blogs, then we would have a huge storehouse of information at our disposal which will be very useful to the people. This is where the opinion mining problem comes up. This task also deals effectively with another aspect of information retrieval i.e. summarization of the information available. The query-focused summarization is useful in presenting the user with a compact answer to what exactly s/he wants instead of giving an entire document/s to the user as output.

2 Background

DUC (precursor of TAC) has conducted competitions on summarization, but the TAC 2008 task on opinion summarization of blogs is a relatively new research field. A number of techniques have already been developed for automated document summarization [5], both single and multi-document like SUMMONS (SUMMarizing Online NewS articles) [4]. Graph-based models such as PageRank, LexRank [3] have been used for ranking documents and sentences, and summaries developed on the node-weights and their similarities. Experiments have been going

on for developing good Opinion extraction systems. The kind of data available on web like, product reviews, blogs, movie reviews, surveys etc. have facilitated the need to develop a system which can cater to a user's query on the opinions expressed in any document on any domain. For example, a consumer who wants to buy any product, say a camera, would be very happy to get a quick summary of the opinions expressed by other consumers about any particular camera model or a more general overview of consumer reviews on different camera models. Also the opinions extracted should be summarized and presented in a structured format which is easy to understand for the users.

3 Our Approach

Our entire system was divided into three main modules: Text extraction, Opinion sentence extraction and summarization. The major steps in that are as follows:

1. First, a simple parser was developed for extraction of relevant text from the blog documents provided. This module gave us all the sentences present in the blog documents (Let's call the set of these sentences to be D).
2. The sentences extracted from the blog documents (D) were then used by the Opinion extraction module to extract all the sentiment sentences (let it be set S, $S \subseteq D$). This module also made use of the target query and optional snippets provided by TAC along with a sentiment wordlist acquired from a corpus (sub-section 3.1). Firstly, we filtered out the query-related sentences from D, and then opinion sentences(S) were extracted from these filtered sentences. In case we could not get any opinion sentences due to inefficient query-based text filtering or opinion extraction, then we used the snippets to derive the opinion sentences from that document. Then we compared the length of S with the summary word-limit.
3. If the opinion sentences(S), so extracted, were within the word-limit of the summary then these sentences were reordered as per their document index keeping the sentences in the same document together. This was done to make the summary more fluent making a crude assumption that two sentences in the same document are more likely to be coherent.
4. If the extracted opinion sentences(S) were not within the summary word-limit, then we used the extractive summarization module to get the final opinion summary. The summarizer used only the opinion sentences(S) for generating the summary, ignoring all other text in the blog documents.

We present below an overview of our system, depicting the modular structure that we have implemented.

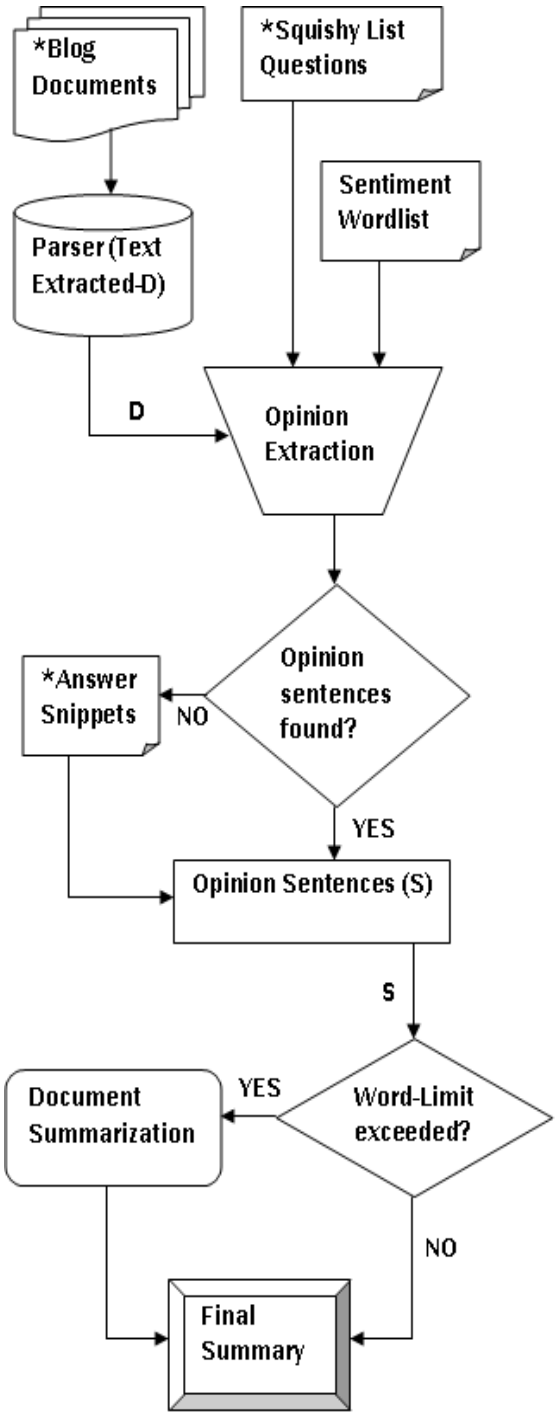

Figure 1: *An overview of the System*. (* → *provided by TAC*)

The algorithm for opinion extraction uses a sentiment wordlist to determine the opinion polarity of a text and based on a ranking formula determines the opinion content of a piece of text. We used an extractive summarization system based on the Lexrank algorithm for summarization but with a new formula for ranking of the sentences. The entire system was built in Python and we used the Natural Language ToolKit extensively. The sub-section 3.1 presents the method used to acquire the opinion wordlist. In sub-section 3.2 we explain the method used for Opinion extraction. And in sub-section 3.3 we show the algorithm used for document summarization.

3.1 Vocabulary Acquisition

For our system, we needed a list of sentiment words. For this purpose, we used seeding. At first we analyzed some documents on the web, mostly blogs to collect a set of approximately 100 positive and 100 negative opinion words. The blogs that we analyzed mainly dealt with movie reviews, political reviews and book reviews. We also made extensive use of the opinion corpus by Bo Pang and Lilian Lee [1] for finding the list of sentiment words. After manually preparing the initial seed list, we passed the list through the Wordnet. During extraction from Wordnet, we have considered only Synonyms, Hyponyms, Meronyms for same polarity and the Antonyms for words

with opposite polarity. The results that we obtained were as follows:-

Number of Iteration	Number of words obtained	Number of words correctly classified	Percentage accuracy
First	1634	1458	89.22%
Second	3872	3109	80.29%
Third	6982	6108	87.48%

Table 1: Results of sentiment words extraction from corpus

The corpus acquired by the above method consisted of two lists of words – one for positive opinion polarity and the other for negative opinion polarity. The words were then manually processed to correct the errors caused during the automatic classification process to get a more refined list of words.

In the next step, the words were classified according to sentiment level. This part of the operation was also manual. We grouped the words under the following categories:-

- Strongly Positive (Score=2)
- Weakly Positive (Score=1)
- Mixed opinion (Score=0)
- Weakly Negative (Score=-1)
- Strongly Negative (Score=-2)

The final corpus consisted of a list of sentiment words that were grouped under the polarity and the strength of opinion. We considered unigrams and bigrams only in this system because of the generic nature of the system. A more domain specific system would be able to incorporate higher order N-grams in its computational model.

Another part of the corpus was a list of some 50 quantifiers that are able to reverse the opinion polarity of a word. For example:-

This is a good book.

This sentence conveys a positive opinion. But the following sentence:-

This is not a good book.

conveys a negative opinion. This is due to the presence of the word '*not*' which reverses the polarity of '*good*'. This list is an integral part of the opinion extraction algorithm.

3.2 Algorithm for Opinion Extraction

The opinion extraction algorithm is run on the individual documents belonging to a particular topic separately [2]. It has not been designed for multi-document opinion extraction. Another feature of this system is that it is query-focused. The opinion finding algorithm consists of the following steps:-

STEP 1: The query statement is parsed to extract the names in the query statement. These names form the search keywords. Also, the words in the query statement are matched with the list of sentiment words in order to find out what kind of opinion is being asked for by the user. In case of no

matches, the words are passed through Wordnet and the resulting set of Synonyms, Meronyms and Hyponyms are matched with the words in the existing wordlist.

STEP 2: Segment the document into sentences.

STEP 3: For each sentence, find the occurrence of the query keywords (say *w*). Let they be located at position *i* in the sentence.

STEP 4: Check for all words in positions *(i-6)* to *(i+6)*. If there is a sentiment word in that region, mark that sentence as one opining about word *w*. If not, go to step 2.

STEP 5: If the sentiment word occurs at position k, check for words in the positions *(k-2)* to *(k+2)*. If there are words in that range which are also in the list of negation words, then reverse the polarity of the sentence.

STEP 6: From the list of polarity strength, calculate the total strength of the sentiment words. This computation is to be done keeping in mind the reversal of polarity. For example, a word with strength -1, if reversed, gets a strength of +1. Compute the average polarity of words by dividing the sum by the number of sentiment words found.

STEP 7: The opinion polarity of the sentences is computed as follows:-

- Average Polarity>1 => Strongly positive opinion
- 0.3<Average Polarity <1 => Weakly positive opinion
- -0.3<Average Polarity <0.3 => Neutral opinion
- -1<Average Polarity <-0.3 => Weakly negative opinion
- Average Polarity <-1 => Strongly negative opinion

STEP 8: Goto step 2

The proximity from the keywords for searching for sentiment words has been determined experimentally. We performed a number of experiments on the distance from the keywords and found that a distance of 6 on either side gave the best results, both in terms of precision and recall.

3.3 Summarization of Extracted Opinions

The list of opinion sentences extracted from the documents was next passed through the summarization system. The system is based on single document extractive summarization. We did not perform any Natural Language Generation. The basic algorithm used for summarization task acts on the term frequency list and tries to identify sentences with maximum weights (according to the formula we used) and groups the maximally dissimilar sentences into the summary. We have done a simple unigram matching of relevant words (nouns, verbs) to find similarity between

any two sentences. Thus the summarizer also makes sure that there is no repetition of information in the final summary.
The algorithm is as follows:-

1. We prepare the Word Frequency list for the document without considering the stopwords.
2. Following the following rules get a new document from the old one:-
   a. Find the noun and pronoun and link them first, if compatible, or else move back along the document to get a proper matching noun.
   b. For a text within quotes, 'I' refers to the last personal noun before 'say','told','said' etc.
3. Each sentence was given a weight based on the Word Frequency list. A threshold value was experimentally determined and all words with frequency above the threshold were considered determining the sentence weights.
   For a sentence S, Weight=W(S) $=\frac{1}{n}\sum_{i=1}^{n} w_i$, where w_i=Frequency of term above the threshold.
4. We find the similarity between two sentences using a simple unigram matching. We defined a value Relation Coefficient (RC) to show the relative similarity between any two sentences. RC was defined as (Number of unigrams matched / maximum number of unigrams in any one of the two sentences matched).
   Example:
   S_1= My name is Tom Sawyer.
   S_2=Tom is friends with Huck Finn.
   Here the matched unigrams are 'Tom' and 'is' .Length of S_1=5. Length of S_2=6.
   So, RC_{12}= (2/6) =0.3333
5. We take the sentence with the highest weight. Let it be S_i. S_i is added to the summary list and removed from the sentence list. To reduce redundancy we removed any other sentence (S_j) with a RC value>=0.5 with S_i.
6. If the sentence list does not become empty, we repeat from step #5
7. We repeat step #1 to step #6, till we get the summary limit.
8. Finally all the sentences in the summary list are rearranged according to their indexes in the original document to get the final summary.

This algorithm takes into consideration the information content in a sentence by the Term Frequency measure and also makes sure that there is no repetition of information by computing the RCs between the sentence pairs and selecting only the dissimilar sentences. In the end all the sentences in the summary are restructured on the basis of their index in the document with the assumption that this will keep the overall coherence intact to some level. It has $O(n^2)$ space complexity and a time complexity of $O(n^3)$ where n = The number of sentences in the document to be summarized.

4 Evaluation

In the TAC 2008 Opinion Summarization task, we were provided with blog documents, "squishy list" questions, and optional answer snippets from the TAC 2008 Opinion QA task. For evaluation, TAC considered summaries for 22 targets for each team run. Finally there were 36 evaluated runs. We present below brief information of the TAC Results for our run and it's comparison with other teams' performance.

Score Category	Score for Our Run	Highest Score among 36 evaluated runs	Position/Rank of our run among the 36 evaluated runs
Average Pyramid F-score (beta=1)	0.332	0.534	8th
Average score for Grammaticality	5.545	7.545	12th
Average score for Non-redundancy	6.045	8.045	16th
Average score for Structure/Coherence	2.636	3.591	20th
Average score for Overall fluency/readability	3.909	5.318	10th
Average score for Overall responsiveness	4.364	5.773	9th

Table 2: Summary of Official TAC 2008 results

5 Conclusion And Future Work

We submitted a single run for TAC 2008 opinion summarization task. Our system had two separate modules for opinion extraction and summarization. The TAC 2008 results for our system reflect that without proper implementation of sentence ordering, the coherence and grammar of the summary is not good. We used the rearrangement of the sentences as per the indexes in the original document to improve the fluency to certain extent. But this did not give us good results for coherence. A more extensive handling of

sentiment analysis can improve the overall accuracy of the summary by improving its precision.

We also found that for some target documents, our opinion extraction module returned opinion sentences well within the summary-limit. In such cases we need not go for any further summarization and instead restructuring the opinion sentences gave the final summary. This was illustrated in the summary for targetID 1001.

Since this is the first time we are participating in TAC, we were unable to try more sophisticated and efficient techniques. In future work, we plan to experiment with new methods for opinion extraction, and extended co-referencing to improve the structure of summary. We will also try to work on improving the readability with proper sentence formation in summary.

Acknowledgements

We would like to express our sincerest gratitude to Dr. Sudeshna Sarkar (Professor, Dept. of Comp. Sc, IIT Kharagpur) for her constant support and invaluable guidance which has been of great help in our endeavors.

References

[1] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan, Thumbs up? Sentiment Classification using Machine Learning Techniques, Proceedings of EMNLP 2002

[2] Bo Pang and Lillian Lee, Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, Proceedings of ACL 2005

[3] Dragomir R. Radev and Güneş Erkan 2004. LexRank: Graph-based Lexical Centrality as Salience inText Summarization.

[4] Dragomir R. Radev and Kathy McKeown. 1998. Generating natural language summaries from multiple online sources. *Computational Linguistics*.

[5] Hovy, E.H. 2005. Automated Text Summarization. In R. Mitkov (ed), *The Oxford Handbook of Computational Linguistics*, pp. 583–598. Oxford: Oxford University Press.