FST-Based Natural Language Processing Method for Opinion Extraction

Delin Liu[1, a], Haopeng Chen[2, b]

[1]Shanghai Jiao Tong University

[2]Shanghai Jiao Tong University

[a]email: sheng2010ren@sjtu.edu.cn, [b]email:chen-hp@sjtu.edu.cn

Keywords: NLP; Rule; Syntax Tree

Abstract. This paper proposes a rule-based and Finite State Transducers (FST) based NLP method for extracting information from massive text. The method differs from n-gram based popular method which relies on probability statistics and machine learning. In our method, the rules are grammars of a language, summarized by people. FST is the implementation tool of rules. It can process natural language and generate a syntax tree for each sentence. To support applying the rules, we tokenize and generate the stem of words, and find many word features which are recorded in a dictionary. After generating a syntax tree, we extract useful information on many aspects, such as subject-verb-object matches and opinion matches. We evaluate our system on the accuracy rate of the syntax trees, and show that the result is satisfactory.

Introduction

The amount of natural language content in Internet raises rapidly. It is too huge for us to handle the, so using computer to process natural language is a good idea.

Understanding the natural language is hard for computer, especially in opinion extraction area. After half a century of progress, people developed two main NLP methods of natural language processing. They are the rule-based method and the statistic based method.

This paper does not focus on the statistic-based method. We propose a rule-based and FST-based NLP method for extracting opinions from massive pure text. This method includes lots of rules which can describe the grammar of a sentence. When processing a sentence, the program matches the sentences with rules, and generates syntax tree of each sentence. Then we can extract opinions from syntax tree. Although the rule-based method is not popular today, it also has many advantages over the statisticbased one. The rule-based method can exactly process every sentence according to the grammars we defined, because the rule is designed to match only one sentence. If we are not satisfied with the processing results, we can adjust the rule grammars easily and get the better result. So it is friendly to programmers to debug and improve it. It does not need any corpus to train itself. The opinion extracted for sentence is accurately, we know who has which opinion on what exactly.

Many researchers are doing some NLP opinion extraction researches in small areas, such as extracting information just from online video comments or from classification of product reviews. Amir Zadeh etc proposed an analysis method for mircro-opinion extracting from online video comments[1]. Pierre brun etc proposed a novel technique[2]. Jihene Jmal etc proposed a feature-based opinion summarization method for extracting opinions in Twitter and SentiWordNet[3]. A new method called deep learning or deep semantic analysis becomes popular these days[4]. Macro Bonzanini proposed a knowledge-based method for summarizing opinions[5]. Richard Johansson proposed a systems that identify opinion expressions and assigns polarities to the extracted expressions[6]. Qingliang Miao proposed a system that combines product feature and opinion extraction for customer reviews by using probalibistic models[7]. Stefan Gindl proposed an approach that combines aspect extraction with opinion target extraction using syntactic patterns[8]. Lakisha L. Simmons proposed a method for extracting financial information from online business reports[9]. Zhu Zhang proposed a scoring method of product reviews[10]. Bing Liu proposed a system to analyze and compare consumer opinions of competing products[11]. Eivind Bjokelund studied opinion extraction and visualization of hotel reviews[12].

We introduce the architecture of the NLP system. Then we introduce the detail method of NLP-processing which is the most important part of the paper, including how to split words, how to generate syntax tree of a sentence and how to make it easier and more efficient to handle some cases. We do some experiments to show our system is accurate and satisfactory.

Design of the NLP System

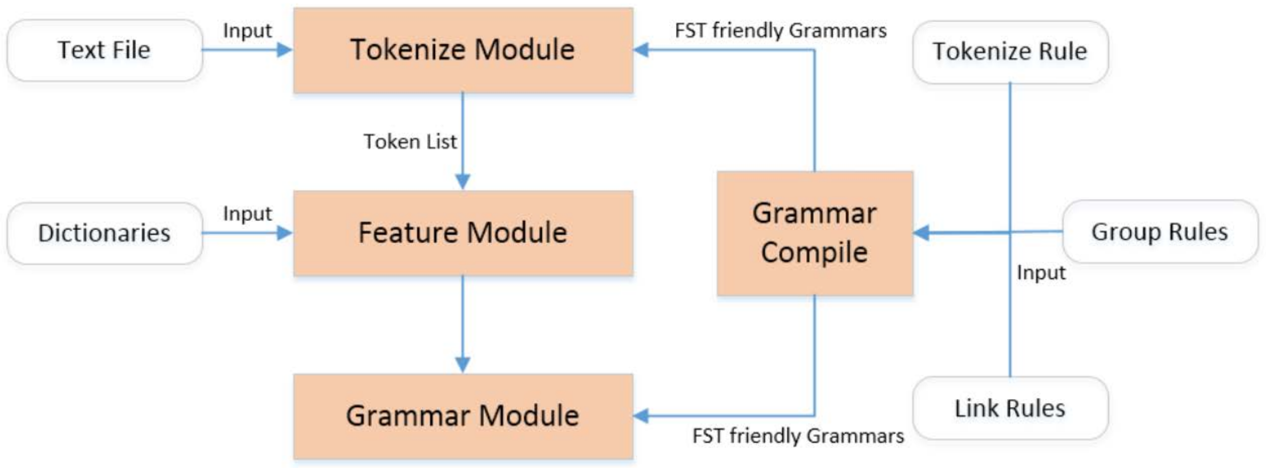

Fig.1. NLP System Architecture

Figure 1 shows the architecture of our system in one single node. The NLP system is used to analyze the the natural anguage and generate a syntax tree of every sentence, and create some opinion links or emotion links between some tokens.

There are four main modules in NLP system: Grammar Compiler, Tokenize Module, Feature Module, Grammar Module.

Grammar Compiler gets the grammars(rules) written by human as input, then checks if they are right, and converts them to FST friendly grammar. Tokenize grammar and rule grammar are written in a programming language which is similar to regular expression and flex bison languages. We extend it to and add some new features we need. The language is friendly to human so we can develop rules easily.

Tokenize Module gets the FST friendly grammar and text file as input, and outputs a list of tokens. The text file contains the natural sentences we want to handle. The module does not split the sentences by space. It splits them by the tokenization FST. The output is a token list. A token means a word, a punctuation, or any other character list.

Feature module contains three submodules, including POSTagging module, normalize module and opinion feature module. POSTagging module accepts a token list, a POSTagging dictionary and POSTagging rules, and find features in a dictionary for each token. Then the module applies POSTagging rules. It picks just the most suitable feature for each token and remove other features. Normalize module accepts a token list and a stem dictionary. It finds a stem in dictionary for each token. We can use the normal format to match the token in later rules. Although normalize module is very simple, it is also an important module in NLP system. Opinion feature module accepts a token list and an opinion dictionary. The opinion dictionary contains lots of words (in normal format). Each word has many opinion features such as ‘love’, ‘dislike’, ‘good’ and ‘bad’. The module attaches these features to the token. In the last step of processing language, we add relations between tokens which are called links. Links are created by checking opinion features. Then we can extract opinions from the links.

Grammar Module accepts FST friendly rule grammars and a token (or group) list with features, and outputs the syntax tree and token links. The grammar module is the core module in our system. It contains many submodules. All of them are used to apply rule grammars.

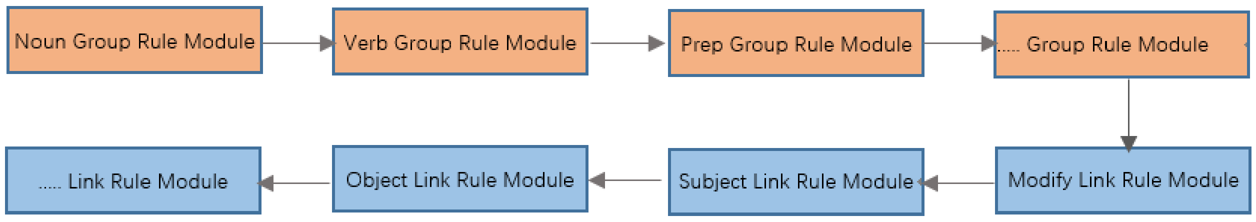

Fig.2. Submodules of Grammar Module

Figure 2 shows the submodules of the grammar module. Besides verb, noun, prep grammar modules, there are many similar modules. At first, the input of verb grammar module is pure token list. The output of it contains groups. A group is a combination of several tokens (or groups). The more grammar modules process the token list, the more groups are in the list. In the end, the token list may become a group list. So we call it syntax tree. The group rule modules process the token list, and create many groups. Finally, we get the syntax tree. Then syntax tree will be passed to link rule modules. Many relations(links) will be added between two tokens or groups. We extract opinions directly from links.

Rules

Rules are the most important part of the system. Rules are usually developed or summarized by linguists (rule developers). A good rule can match many sentences with high accuracy. Our system contains three types of rules: tokenize rule, group rule and link rule.

1) Tokenize rule: It’s not a good idea to split words just by space. One reason is fixed collocations are common in English. We treat fixed collocation as just one word and this will help us a lot later in grammar module. What’s more, a title of a book or film is also one word. The name of a street or city is the same. Another reason is that we cannot expand our system to other languages which don’t take space as separator.

2) Group rule: Group rules are the core rules of our system. Lots of group rules exist in the system to generate syntax trees of sentences.

Applying these rules, we can chunk tokens into groups. At first, a sentence is made up of many tokens. Then some tokens may match one group rule, and become a group after applying the rule. With so much group rules, we can easily make tokens a group. In the end, one sentence just contains one single group. In this group, the verb token is the header and subject and object token is footer.

These are some simple examples of group rules:

be-word = is|are|was|were|be|am

be-going-to = @be-word going to

be-going-to-do = @be-going-to T(verb) T(noun)

With these three simple rules, we can catch many sentences and chunk some tokens into group. Sentences like ‘I am going to finish it’ and ‘He was going to watch television’, will become ‘I [[am going to] finish it]’ and ‘He [[was going to] watch television]’.

It works! But how about ‘I am going to ask him to finish it’? It becomes ‘I [[am going to] ask him] to finish it’. This is not what we want. The syntax tree will go wrong if this rule matches the sentence. The right chunk result is like this ‘I [am going to [ask him to [finish it]]]’.

Therefore, we can see that a single rule can chunk tokens rightly, but it may also lead to some wrong results. We all want to write rules with more right chunks and less wrong results. If the baseline only contains thousands or even millions of sentences, we may miss the wrong results and take a bad rule into our system. So a huge baseline is needed to judge if a rule is good enough to join our rules family.

3) Link rule: Link rule is different from group rule. A group rule describes a token sequence that can make a group. It just means syntax structure. A link rule describes two tokens which have relations between them.

We often put link rules behind group rules in the pipeline, because group rules make groups and generate syntax tree. With a good syntax tree, we can find relations easily within the sentence.

Link rules are less important than group rules. A little change in group rules may totally mess the syntax tree of many sentences. Further more, link result will also change largely because it depends on syntax tree.

Experiment results

Table.1. Result of Experiment 1

All amount	4,322,161
Positive amount	624,528 (14.45%)
Negative amount	402,313 (9.31%)
Sample amount	10,000
Positive amount of sample	1,464
Negative amount of sample	893
Real positive amount of sample	2,357
Real negative amount of sample	1,326
Sample Positive accuracy	62.11%
Sample Negative accuracy	67.35%
Sample Accuracy	64.00%

Table.2. Result of Experiment 2

All amount	4,322,161
Positive amount	492,350 (11.39%)
Negative amount	306,824 (7.10%)
Sample2 amount	10,000
Positive amount of new sample2	1,173
Negative amount of new sample2	631
Real positive amount of new sample2	1,458
Real negative amount of new sample2	793
Sample2 Positive accuracy	80.45%
Sample2 Negative accuracy	79.57%
Sample2 Accuracy	80.14%

We crawled about 4.3 million sentences from twitter, and put them into our system as input. At first we get an unsatisfactory result, because our system is not aimed at twitter. Then we add dozens specific rules for twitter, and the result is satisfactory.

We assess our system by the accuracy, but we have less time to check all 4.3 million sentences, so we randomly draw 10,000 sentences from them as sample. We check if the sentences are right with the result manually, and get the accuracy rate. Table 1 is the result of the first experiment. We input all 4,322,161 sentences. According to our system we find 14.45% of them contain positive opinions, 9.31% contain negative opinions. We draw 10,000 sentences as sample. The positive and negative opinion amounts among sample are 1,464 and 893 according to our system. Then we check these 10,000 sentences manually, and find the accuracies are 62.11% and 67.35% for positive and negative opinions. Therefore, the total accuracy is 64.00%. This accuracy is low and unsatisfactory. So we add dozens specific rules just for twitter.

Table 2 is the result of the second experiment. We improve our system by adding specific rules, and the result is satisfactory. We can see that our system can be improved easily by adding some grammar rules. The effect of adding specific rules is satisfactory.

Conclusion

The rule-based NLP method can be equal to opinion extraction. The core idea of it is rules. We can make the system better by adding rules. Tokenizing and POS-Tagging are also important parts of the systems. The system can reach a relatively low accuracy when it is not aimed at the input. But it can be improved easily by adding specific rules. Therefore, the system can be used in various fields, and just need some targeted rules.

References

[1] Amir Zadeh, Micro-opinion Sentiment Intensity Analysis and Summarization in Online Videos, Multimodal Interaction, Pages 587-591, 2015.

[2] Henri Avancini, Alberto Lavelli, Fabrizio Sebastiani and Roberto Zanoli, Automatic expansion of domain-specific lexicons by term categorization, Transactions on Speech and Language Processing Volume 3 Issue 1, Pages 1-30, 2006.

[3] Klaus Berberich and Srikanta Bedathur, Computing n-gram statistics in MapReduce, Extending Database Technology, Pages 101-112, 2013.

[4] Eivind Bjorkelund, Thomas H. Burnett and Kjetil Norvag, A study of opinion mining and visualization of hotel reviews, Information Integration and Web-based Applications & Services, Pages 229-238, 2012.

[5] Marco Bonzanini, A knowledge-based approach for summarising opinions, Research and development in information retrieval, Pages 991-991, 2012.

[6] Pierre Brun, Hideki Kawai, Kazuo Kunieda and Keiji Yamada, ChronoSeeker: Future Opinion Extraction, Web Intelligence and Intelligent Agent Technology - Volume 01, Pages 568-571, 2009.

[7] Jonathan D. Cohen, Recursive hashing functions for n-grams, Transactions on Information Systems Volume 15 Issue 3, Pages 291-320, 1997.

[8] Florian Wogenstein, Johannes Drescher, Dirk Reinel, Sven Rill and Jorg Scheidt, Evaluation of an algorithm for aspectbased opinion mining using a lexicon-based approach, Issues of Sentiment Discovery and Opinion Mining, 2013.

[9] Stefan Gindl, Albert Weichselbraun and Arno Scharl, Rulebased opinion target and aspect extraction to acquire affective knowledge, World Wide Web, Pages 557-564, 2013.

[10] Alexander Hogenboom, Daniella Bal, Flavius Frasincar, Malissa Bal, Franciska de Jong and Uzay Kaymak, Exploiting emoticons in sentiment analysis, Applied Computing, Pages 703-710, 2013.

[11] Jihene Jmal and Rim Faiz, Customer review summarization approach using Twitter and SentiWordNet, Web Intelligence, Mining and Semantics, Article No. 33, 2013.

[12] Richard Johansson and Alessandro Moschitti, Extracting opinion expressions and their polarities: exploration of pipelines and joint models, the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2, Pages 101-106, 2011