Technical Review
A Comparison of Sentiment Analysis between CoreNLP vs LingPipe

Introduction

Both CoreNLP and LingPipe are Java based toolkits used for text processing and analysis that can be used to perform sentiment analysis. Sentiment analysis seeks to identify and extracts the subjective content, which is the underlying opinion, of a particular text span. An example of sentiment analysis is to classify if a movie review is positive or negative. LingPipe provides APIs to complete common text mining tasks such as tokenizing, feature extracting, and classifying. LingPipe uses a mixture of heuristic rule-based and statistical components. CoreNLP is able to perform similar tasks. This toolkit was built more recently and utilizes rule-based, probabilistic ML and deep learning components. LingPipe and CoreNLP use different underlying algorithms and models to achieve the classification of text based on sentiments.

Algorithm Comparison

LingPipe

We can treat sentiment analysis as a special case of categorization problem, where there are only two categories of text: positive and negative. There are many approaches to this classification problem. We can use language model classifiers in LingPipe to classify each review / document by polarity and classify each sentence as subjective or objective. For example, logistic regression classifier is commonly used to classify sentences. One recommended approach to classification using LingPipe is the hierarchical polarity analysis, which was introduced by Pang and Lee in 2004. In basic polarity classification, standard ML classification techniques such as SVM is used to directly categorize text at document level. This hierarchical approach improves the result by first removing objective sentences from the text and then classifying the remaining subjective sentences. This technique helps reduce noise in the sentences. When the input texts get fed into the classifier, LingPipe reduces the sentences to top N most subjective (opinionated) sentences ranked by the subjectivity model. The subjectivity classifier model computes the likelihood of a sentence being subjective by assuming adjacent sentences have similar objective-subjective labels and we can use the min-cut algorithm to classify sentences (Bo, 2004).

To implement, we can instantiate a `Classification` class and supply labeled training data with the handle method.

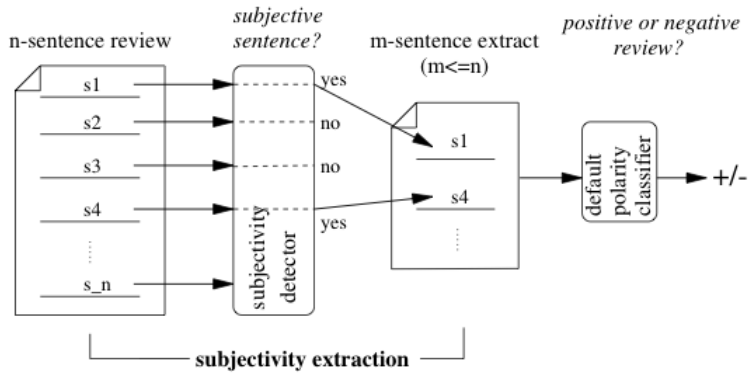
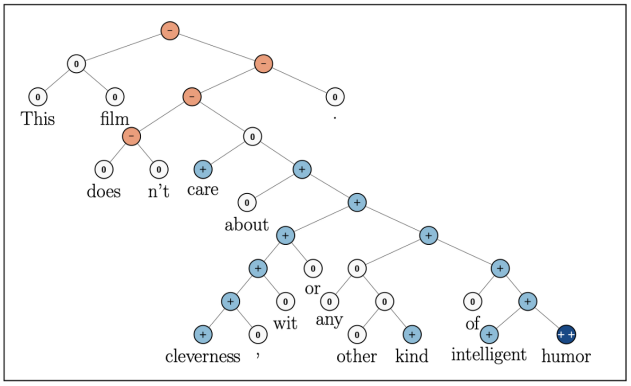


Figure 1: Polarity classification via subjectivity detection.

CoreNLP

Sentiment analysis can be particularly challenging when there's negation like "not bad" or sarcasm in a sentence. Classifiers will likely place them in the wrong category if they just look for words that are positive or negative. The Stanford Sentiment Treebank (SST) compiles data by splitting reviews into phrases and asks annotators to label random N-gram text into five different categories from very negative to very positive. SST parses sentences to "sentiment treebanks", which allows classifiers to consider the sentence structures when doing classifications. At each node or level of the tree, a five-category score is computed for the result at the current node level. The results are then passed to the parent nodes and a new score is to be computed at the parent level. We call this model the Recursive Neural Tensor Network (RNTN). It computes the root node's sentiment using a bottom-up approach where each level of parent nodes computes the score from the child nodes using the same tensor-based composition function (Socher, 2013). When comparing this approach with baseline classifiers like Naive Bayes and SVM, we observe significant improvement in accuracy.

To use CoreNLP to perform sentiment analysis, we just need to use the **`nlp.annotate()`** function and specify 'sentiment' as one of the parameters to annotate. The SentimentAnnotator implements Socher's sentiment model.



In conclusion, both LingPipe and CoreNLP offer similar set of tools which we can use to perform linguistic analysis such as tokenization, entity extraction, POS tagging, and classification models. They both provide a set of APIs which hide the implementation details which we can use conveniently. CoreNLP contains pre trained deep learning models which can be used directly for sentiment analysis, and training is only required if the interested text belongs to a particular subdomain, while LingPipes uses mostly classic conditional probabilistic LMs and would require users to train data with custom labeled text before using the classifiers.

Reference

alias-i. (n.d.). *LingPipe: Sentiment Analysis Tutorial*. LingPipe.

http://www.alias-i.com/lingpipe/demos/tutorial/sentiment/read-me.html

Bo Pang and Lillian Lee. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. ACL Proceedings.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP Natural Language Processing Toolkit. *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Published.

https://doi.org/10.3115/v1/p14-5010

Socher, R., et al. (2013, October). *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*. Empirical Methods in Natural Language Processing, Seattle, Washington, USA.