Sentiment Classification on Syntactically Parsed Short Documents

Tyler Trine

Computational Semantics
Final Project Report

Introduction

Background and Motivation

Computational semantics has been a large part of machine learning research, but the rise of micro-blogging services, such as Twitter, presents new challenges. Short-form text encourages abbreviation and leads to new grammatical structures.
Inducing a word space is one task, measuring the accuracy with which it represents its domain is another. Sentiment analysis is one way to measure how well an algorithm has been trained on its input data.

Problem Statement

Is sentiment inference significantly more accurate when the word space takes syntactic information into account?

Related Work

This paper draws heavily from the ideas of the paper: Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank (2013). This paper establishes the state-of-the-art in sentiment classification in the domain of movie reviews. The question then becomes, is this classification prowess transferrable to domain-specific short-form text?

Methodology

Data Acquisition, Exploration, and Pre-Processing

While micro-blogging has given researchers access to a vast wealth of information, it has also provided a lot of noise. Many tweets are random and serve no purpose, which can skew results and increase complexity. Like Twitter, StockTwits is a microblogging site. Users use the service to post about the stock market. It has a cashtag, '$', to allow users to refer to stock ticker symbols. This is particularly well-suited to the task of sentiment classification, as cashtags explicitly indicate the topic of the post.

Of the more than 200,000 tweets collected, about 20% of them carried a metadata label reading either "bullish" or "bearish." This provides a wealth of training data for the classifier.

I obtain tweets from the StockTwits site by scraping it directly. Although they expose some of their data via an API, I needed some tweet attributes available only on the website; namely, the

prelabeled sentiment that some tweets come with. Scraping was performed in Python using the popular tool Scrapy.

I collected tweets from the front page of every active user. I also collected the date of posting and the associated sentiment, if any. After filtering out unhelpful tweets, the final dataset size was 217,000.

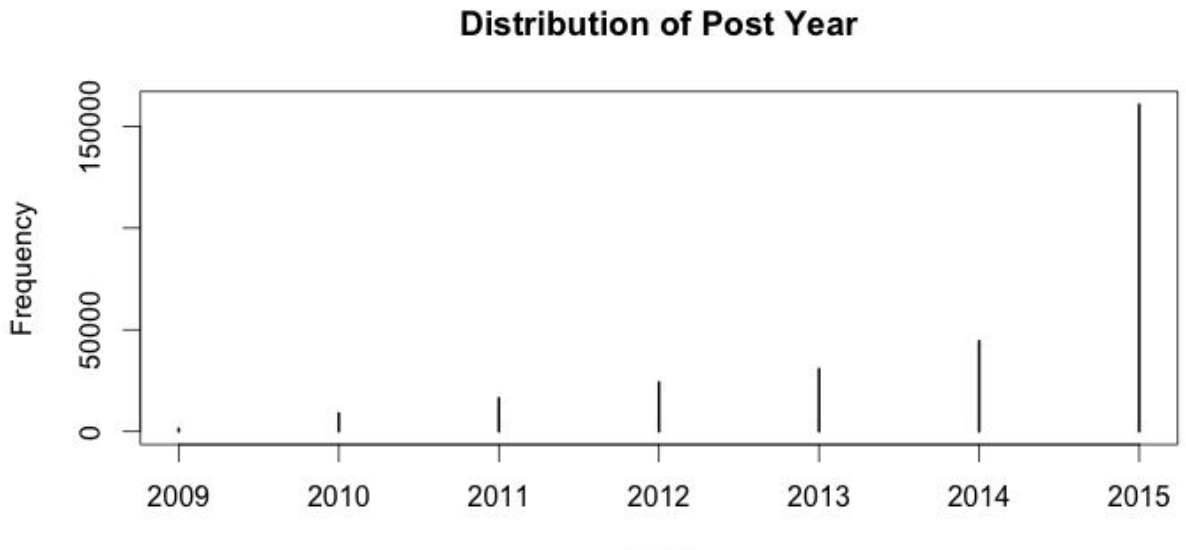

fig. 1

Sentiment Detection

In 2013, researchers at Stanford introduced new compositional distributional semantic methodology known as a Recursive Neural Tensor Network (RNTN). The RNTN captures syntactic information in unstructured text in a deep way. It does this by computing word meanings recursively from the level of the word to the level of the sentence. The original paper performs sentiment analysis on sentences. The code is open source and can be downloaded at http://nlp.stanford.edu/sentiment/index.html. However, as with many sentiment analyzers, it was trained on movie reviews. Investors and finance professionals have their own jargon with little meaning outside the finance world. For example, “bullish”, “bearish”, “long”, “short”, and many more are not part of non-finance vocabulary. In fact, in preliminary tests, the RNTN’s default model incorrectly classified most tweets as having negative sentiment. Thus, to get accurate results, I retrained the model on our new dataset.

This work fell into two stages: Parse the dataset syntactically, and train a model. For the former, I selected the subset of our data which users explicitly annotated with sentiment. This is a feature

on StockTwits. About 20% of our collected tweets have this annotation. To maintain sentiment depth, as well as encourage training results more representative of domain-specific vocabulary, I explicitly mark the sentiment of 19 highly opinionated words such as “bullish,” “bearish,” “short,” “sell,” and “buy.” I obtained this choice of words in the following manner:

- Isolate all tweets which contain a cashtag and one other word.
- Count the frequency of each such unique word, and analyze the distribution of sentiments associated with this unigram.

Figure 4 plots this unigram frequency against the standard error of the user’s sentiment. Standard error can, in this case, be roughly viewed as the amount of disagreement as to the sentiment of the singleton. Words which did not occur frequently tended to be controversial. However, the most common words have no such divergent opinions, indicating that they are good choices to explicitly train the model on. I selected all words occurring more than 20 times in our data.

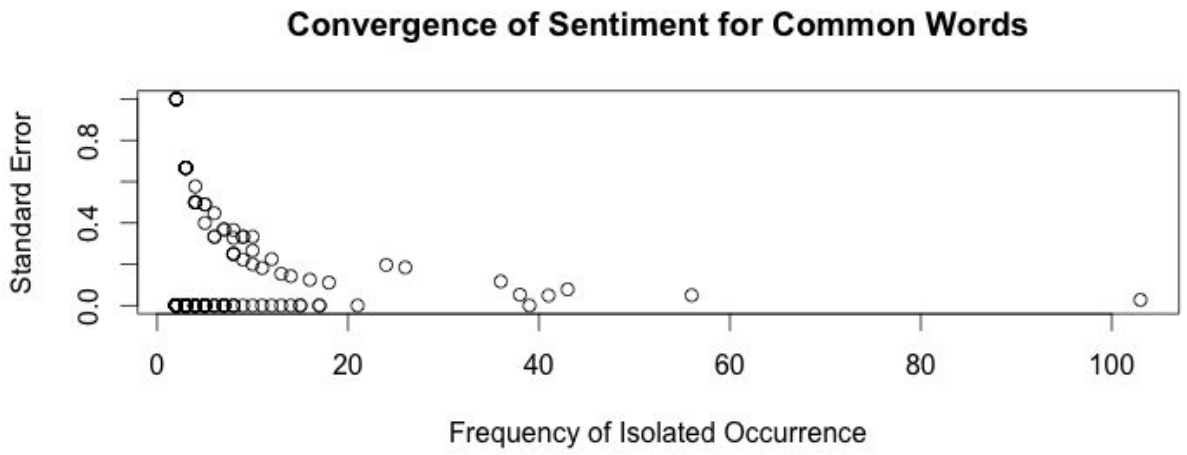

fig. 2

Results

The classifier accurately inferred the sentiment of tweets from the test set 72.4% of the time. According to the cited paper, humans tend to agree just 70% of the time. My attempt to establish a baseline with a Naive Bayes classifier over a ‘flat’ (non-recursive) semantic space is left unfinished. I induced a wordspace on the training data using Random Indexing, however, the Naive Bayes classifier is throwing an unresolved exception when I try to run it. Furthermore, the size of the training and testing datasets are unwieldly (> 500mb of text), and do not readily yield to debugging. I plan to further this progress over break, because I am quite curious if the RNTN is truly better, and if so, to what extent.

Citation

1. Socher, Perelygin et. al., Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank (2013)