SemEval 2015 Task 1: Paraphrase and Semantic Similarity in Twitter

Michael Meding
University Massachusetts Lowell
1 University Ave
Lowell, MA 01854, USA
mikeymeding@gmail.com

Hoanh Nguyen
University Massachusetts Lowell
1 University Ave
Lowell, MA 01854, USA
hoanh.lam.nguyen@gmail.com

Abstract

Hoanh and I decided that we would do the SemEval 2015 Task 1 for our NLP project this semester. This task involves paraphrase and semantic similarity in Twitter which is formalized as follows, Given two sentences, the participants are asked to determine whether they express the same or very similar meaning and optionally a degree score between 0 and 1. Following the literature on paraphrase identification, we evaluate system performance primarily by the F-1 score and Accuracy against human judgements.

1 Introduction

Our first task with this project was to translate the original starting code from Python to Java. This required rewriting both the main logistic regression function to a Hidden Markov Model with no hidden layers and to rewrite the data parser so as to interface with the same data that was given to us for this task.

2 Data

SemEval provides all of its tasks with data for use with evaluating the results of your work. In our prior Twitter project we hand annotated our own data set which was extremely time consuming and frustrating. We also did not have any kind of verification of our data set so the accuracy left much to be desired. Luckily, the data sets which were provided to us for this task are both consistent and accurate. Below is an example of the data provided to us.

The "Sent_1" and "Sent_2" are the two sentences, which are not necessarily full tweets. Tweets were tokenized (thanks to Brendan O'Connor et al.) and split into sentences.

The "Label" column is in a format such like "(1, 4)", which means among 5 votes from Amazon Mechanical turkers only 1 is positive and 4 are negative. We would suggest map them to binary labels as follows: paraphrases: (3, 2) (4, 1) (5, 0) non-paraphrases: (1, 4) (0, 5) debatable: (2, 3) which you may discard if training binary classifier The "Sent_1_tag" and "Sent_2_tag" are the two sentences with part-of-speech and named entity tags (thanks to Alan Ritter).

3 Base Line

The baseline implementation used a logistic regression model and simple lexical features. The features made uses of unigrams, bigrams, and trigrams of the words and the porter stem of the words. It calculates the precision (intersection verses original ngrams), recall (intersection verses candidate ngrams), and the F1 providing a total of 9 features. Our reimplementation of the baseline was able to achieve an F1 score of just over 0.5 on the dev set after training on the training set. This is only slightly worse than the implementation that was provided. The results from the given Python baseline was 0.5 for the F1 score which with later tweeks we managed to get very close to.

4 Data

The data which was provided to us was consisted of two files, a training data set and a development data set. The training data set consisted of 13063 Tweet pairs and the development data set consisted of 4727

Tweet pairs. These data sets were organized as tab separated values categorized as such,

5 Base Line Modification

Our first thought was to see what we could do to improve the baseline. After some time and research we decided to see what would happen if we simply omitted the trend. So now the features would make uses of ngrams before and after the trend. With that change the F1 score on the dev set went up to over 0.54. Seeing this we elected to have all our features use the same approach of omitting the trend.

6 Ark Tweet NLP

Ark Tweet NLP is a part-of-speech tagger that was built specifically for Tweets. In a previous project we created features that looked at ngrams of the tags and those features worked quite well. For this task we followed the idea introduced in the original python baseline and calculated the precision, recall, and F1. The actual F1 score on the dev set was just shy of 0.36.

7 Harvard General Inquirer

The Harvard General Inquirer is a lexical resource that provides a number of categories that a word belongs to. There are 182 categories however most don't show up very often. For our data set the Harvard Inquirer categories that we settled on were those that appeared more than 3000 times in the training set. For features we took a bag of words approach and used the categories found in the original and candidate tweets. Aside from that we again calculated the precision, recall, and F1 of the mutual category count verses the category count for the original and candidate tweets. The F1 score of these features was just under 0.31.

8 SentiWordNet

SentiWordNet is a lexical resource that provides sentiment scores for words. Each SentiWordNet entry contains five explicit attributes (part-of-speech, id, positive score, negative score, synset terms, and glossary) and an implicit attribute (objective which is 1 - positive score - negative score). The intuition behind using sentiment is if one statement has a positive sentiment and the other has a negative sentiment then it would likely not be a paraphrase. The features we implemented were scores divided by entry count, if there are more negative entries than positive entries, scores divided by non-zero score entry count, scores for adjectives divided by non-zero score entry count, and binary features for majority counts. These features looked at each statement individually. Most of these features were inspired by the Opinion Mining Using SentiWordNet paper. Some features that looked at both statements were binary features that check if both had majority score or counts. The F1 score of these features was around 0.06.

Our first experiment after establishing a good baseline was to score the tweets based on word level sentiment. We preformed a crude run with sentiment weighted heavily to see if we could get any kind of result. This unfortunately was quite bad and did not yield an improvement of any kind. We pushed a bit further by attempting to score the entire tweet and getting an average for comparison but the results were equally as bad. During this time we had acquired Willie Boag's Twitter sentiment code from a prior SemEval competition to see if his sentiment analyser could improve on our crude model. However, after seeing the dismal results using only a crude model we decided that it would be of more value to pursue other features to improve our model.

9 MPQA Subjective Lexicon

The MPQA Subjectivity Lexicon contains entries and provide the strength of the subjectivity and polarity. For this resource we created a number of binary features. The features compare the number of negative polarity counts to positive polarity counts and another was designed to compare the total number of weak counts to strong counts and negative counts to positive counts. The F1 score of these features was about 0.13.

10 Wordnet Synonym

Many words have a number of synonyms associated with them. A resource that was able to give us synonyms of a word was Wordnet. So we take all the words, get the synonyms for those words, and put all of it in to a set. Then we calculate the precision, recall, and F1 similar the way the baseline does except we calculate the intersection of the words to the words plus synonyms of the other statement. The F1 score of these features was close to 0.54.

11 Harvard Inquirer with Wordnet Synonym

12 Final Results

Acknowledgments