RJ Walsh

CS224N – Final Project

Sentiment Analysis of Stanford Course Reviews

Introduction

The goal of this project is to investigate the sentiment of Stanford course reviews. Reviews are a common element of the modern web, and determining the sentiment of a review offers great potential into analysis of the items being reviewed. In the domain of Stanford courses, this could offer professors, students, and administrators the ability to qualify course reviews on a new level; giving everyone a new method of determining how students feel about the courses they take. This paper investigates various classification methods to attempt to solve this problem.

Data

The data used for this project was a corpus of 4247 course reviews obtained from CourseRank (http://courserank.stanford.edu). The reviews were obtained as a raw dump from the database, with no additional information given. Almost all the reviews obtained were easily classified, since the purpose of a course review is primarily to convey sentiment or information about a course.

The data was tagged for sentiment manually but a number of student volunteers. Each review was tagged as one of three classes: positive, negative, or neutral. The student volunteers tagged over 1,000 reviews, resulting in 540 positive reviews, 255 neutral reviews, and 205 negative reviews. This data set was divided into a test set and training set in an 80/20 split. This split meant that the training set was composed of 433 positive, 214 neutral, and 153 negative reviews. The test data set was composed of 107 positive, 41 neutral, and 52 negative reviews.

One initial observation of this data is small, but noteworthy. It is satisfying to note that in general, most reviews tend to be positive or neutral, with only a small percent being negative. Thus, we can conclude that overall, Stanford students enjoy their courses.

Since some reviews were classified as neutral, adding a third class, experiments with each classifier were conducted with and without these reviews. See the results sections for each classifier below for a discussion on how this impacted the results.

Related Work

Sentiment analysis is a topic that has been investigated in a variety of forms. Two papers that I used as references were “Movie Review Mining and Summarization” by Zhuang, Jing, and Zhu, and “Thumbs Up?

Sentiment Classification using Machine Learning Techniques" by Pang, Lee, and Vaithyanathan. Both papers investigate sentiment classification in the domain of movie reviews, which offers suggestions on how to translate the prior work to the domain of Stanford course reviews.

Movie reviews often discuss the technical aspects of a movie, ranging from the performance of a specific actor to the technical implementation of the film. Both papers attempt to identify these features in the review in an attempt to improve the classification task. I will later discuss attempts I made to translate these concepts to the domain of course reviews. For instance, where a particular actor might be a great indicator of sentiment in a course review, a professor or TA might be a fantastic indicator of course review sentiment. Identifying these indicators of sentiment was the primary goal of my investigation.

Codebase

For the project, I utilized a number of engineering resources. I borrowed heavily from the code I wrote for Programming Assignment 3, modifying the MaxEnt classifier I built for that assignment to create a version which can classify entire reviews instead of single words. I also reworked the existing classifier to build a Naïve Bayes Classifier smoothed using add-δ smoothing.

I also tested the data set using the Stanford Classifier, in an attempt to compare the features I engineered to the generic features of the Stanford Classifier.

Text Pre-processing

As discussed in Pang and Lee's "Thumbs Up" paper, there are some steps that can improve overall results. The most notable is the processing of NOT. For instance, without this processing step, the classifier assigned the review "Not recommended" to the positive class, because of the high association between "recommended" and the positive class. However, the presence of not inverts this sentiment, and thus we need to train the classifier differently on it.

In addition to pre-processing NOT tokens, removing punctuation also improved the F-score for the classifiers. Though punctuation is often a great indicator of sentiment, it generally cluttered the classification by fragmenting tokens unnecessarily.

Maximum Entropy (custom)

Description

This classifier implements the same model used in PA3. The starter code from that assignment was modified to classify sentences to String labels. The major changes occurred in the loadData, transformData, and extractFeatures methods. These changes ranged from simply removing the List<> structures to deal with sentences, to reworking the feature set for the domain of course reviews.

The classifier itself adheres to the same properties of a maximum entropy classifier. The probability that a review r will be assigned a class c is given by:

$$P(c|r) = \frac{\exp(\sum_i \lambda_i f_i(c,r))}{\sum_{c'} \exp(\sum_i \lambda_i f_i(c',r))}$$

Where the λ_is are the weights learned by the classifier, and the f_is are binary functions, returning 1 if the review contains the feature, and 0 otherwise.

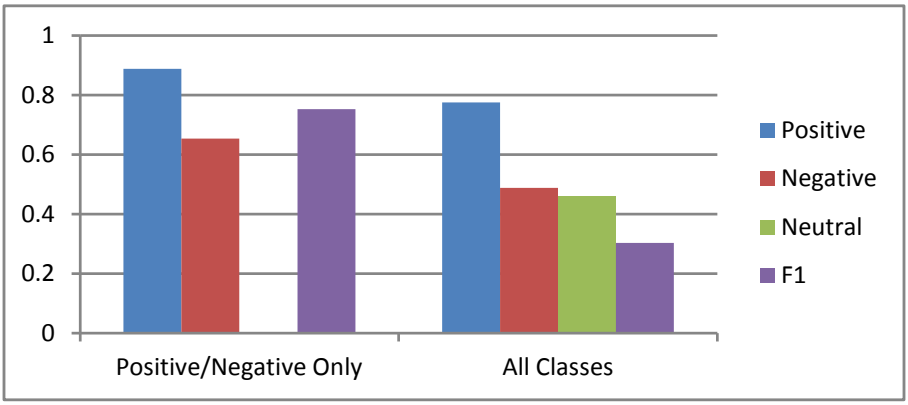
The objective function of the classifier is given by

$$F(\lambda) = -\sum_{c,r} \log\big(P(c|r,\lambda)\big) + \sum_i \frac{\lambda_i^2}{2\sigma^2}$$

Which the maximum entropy classifier attempts to minimize, that is, maximize the total entropy of the classifier.

Results

Data Set	Positive	Negative	Neutral	F1
Positive/Negative Only	0.8879	0.6538	N/A	0.7531
All Classes	0.7757	0.4878	0.4615	0.3037

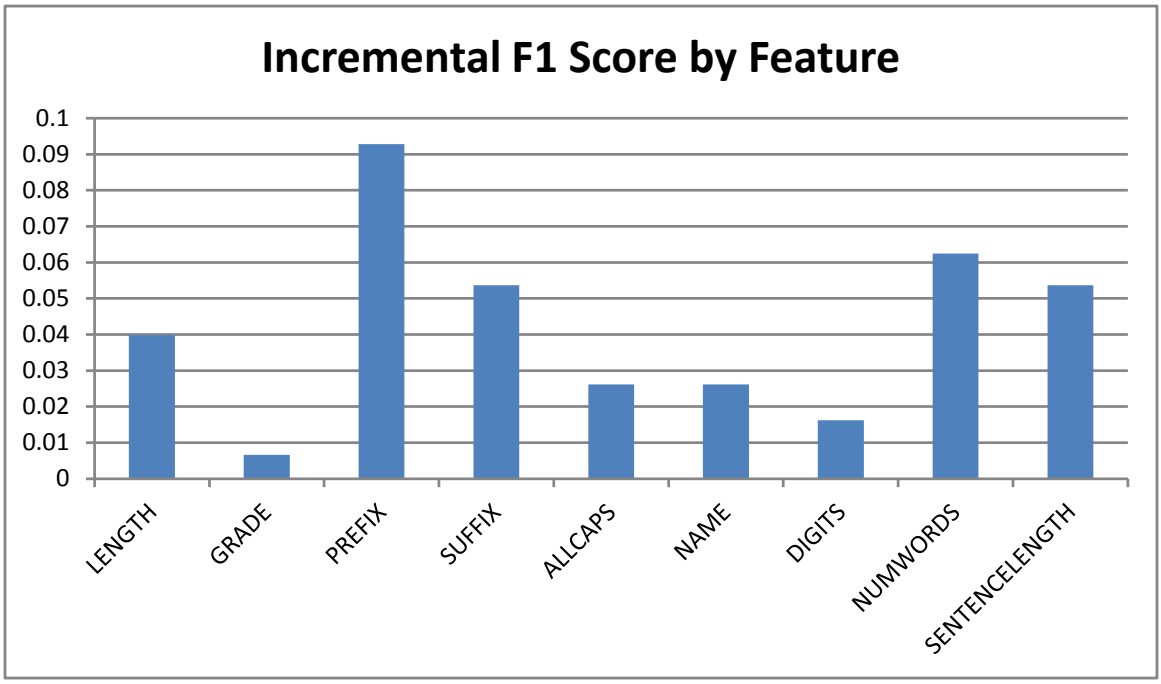


Features

There were a number of features added to the classifier for the task of classifying reviews. These are described below, along with the relative success of each feature on the positive/negative only data set. Each feature is added on a per-word basis, with two per-sentence features added at the bottom of the table. The incremental effect is calculated by determining the F1-score with only that feature removed.

Feature	Description	F1	Incremental F1
WORD	The word itself (baseline feature)	0.7531	N/A
LENGTH	The length of the current word	0.7133	0.0398
GRADE	Whether the current word matches a regular expression for a grade, namely [ABCDF][+-]?	0.7465	0.0066
PREFIX	The first three letters of a word	0.6603	0.0928
SUFFIX	The last three letters of a word	0.6994	0.0537
ALLCAPS	Whether the word is entirely capitalized	0.7269	0.0262
NAME	Is the first letter uppercase? A common trend in names	0.7269	0.0262

DIGITS	Whether the word contains any digits	0.7369	0.0162
NUMWORDS	The number of words in the sentence (per-sentence feature)	0.6906	0.0625
SENTENCELENGTH	The length of the sentence (in words/tokens) divided by 10	0.6994	0.0537



Error Analysis

As indicated above, each feature improved the overall F-Score by a small amount. The features were added incrementally based on the observation of misclassified reviews.

Prefix and Suffix tended to have the largest impact on the overall results. This seems to make sense, as words which convey sentiment in the data set often start with a particular prefix. For instance, "interesting," "reading," and "amazing" appeared often in positive reviews, while "boring," "nothing," and "taking" appeared often in negative reviews. The addition of the suffix as a feature allowed the classifier to better separate these two classes.

The sentence length features had a great combined effect. Longer reviews contain more details about the course, usually neutral and informative. Shorter reviews that express sentiment thus require stronger words to indicate the same sentiment that might be dispersed across a larger, longer review with less powerful words.

The all-caps feature proved to be effective, as words that are in all caps tend to express a high amount of sentiment one way or another. Likewise, the name feature was effective at capturing professor's names in reviews. Reviews that mention a name often contain a higher amount of sentiment, as they have a particular piece of information, be it positive or negative to share about the person. This usually

takes the form of “take this class with Professor A/TA B,” so specifically mentioning a professor can help with classifying the review. As a potential topic to improve the results, classifying the data with pre-process NER could potentially offer a vast improvement over the current results.

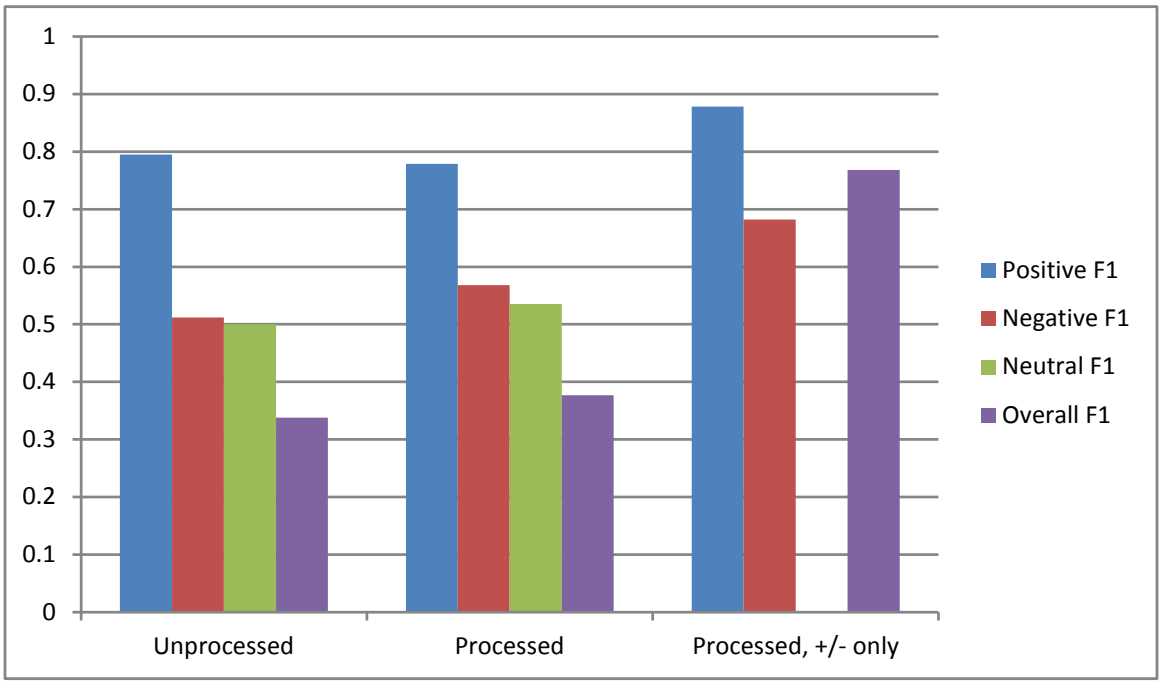
Stanford Classifier

Description

This test utilized the Stanford Classifier as a comparison point on the given data set. I used the Stanford Classifier version 2.0, available at http://nlp.stanford.edu /software/classifier.shtml. The classifier was configured using the same parameters as the cheese/disease data set available in the example configuration. Namely, the classifier uses 1-4grams with prefix/suffix and binned lengths of 10,20, and 30.

Results

Data Set	Positive F1	Negative F1	Neutral F1	F1
Unprocessed	0.795	0.512	0.500	0.338
Processed	0.779	0.568	0.535	0.377
Processed, +/- only	0.878	0.682	N/A	0.768



Error Analysis

The Stanford classifier beats the custom classifier by approximately 0.025 on each F1 score. This is likely due to the use of higher-order N-grams. The Stanford classifier takes advantage of bigrams and trigrams, while the custom classifier does not go beyond unigrams. Despite not having the custom features of the custom classifier, the Stanford classifier still performs well in the general case.

Another noteworthy comparison on the Stanford classifier is the difference between the processed and unprocessed data. The pre-processing techniques improve the overall F1 score, but hurt the unprocessed score. This makes sense, as the pre-processing is largely an attempt to generalize the data by removing specific punctuation. In addition, the NOT_ tokenizing that occurs during pre-processing causes reviews that were previously misclassified (i.e., "Not recommended") to now be classified correctly, but likely has an adverse effect on positive reviews containing not that are incorrectly assigned to the negative sentiment class.

Naïve Bayes

Description

The Naïve Bayes classifier utilizes a simple independence assumption to drastically reduce the complexity of the classifier. From a pure engineering standpoint, the Naïve Bayes classifier file only uses 150 lines of code where the MaxEnt classifier is over 600 lines. The classifier calculates the probability that a review r will be assigned a class c as follows

$$P(c|r) = \frac{P(c)P(r|c)}{P(r)}$$

Since we are only concerned with $\arg\max_c P(c|r)$, we can simplify the equation.

$$P(c|r) = P(c)P(r|c)$$

We calculate $P(r|c)$ using the simplifying assumption of independence, where

$$P(r|c) = \prod_{i=1}^{n} P(r_i|c)$$

Which can be calculated directly from the training data. We can simplify this to a summation by taking the log, and since we can still maximize, this will be correct. Thus, the final computation is

$$\log P(r|c) = \log P(c) + \sum_{i=1}^{n} \log P(r_i|c)$$

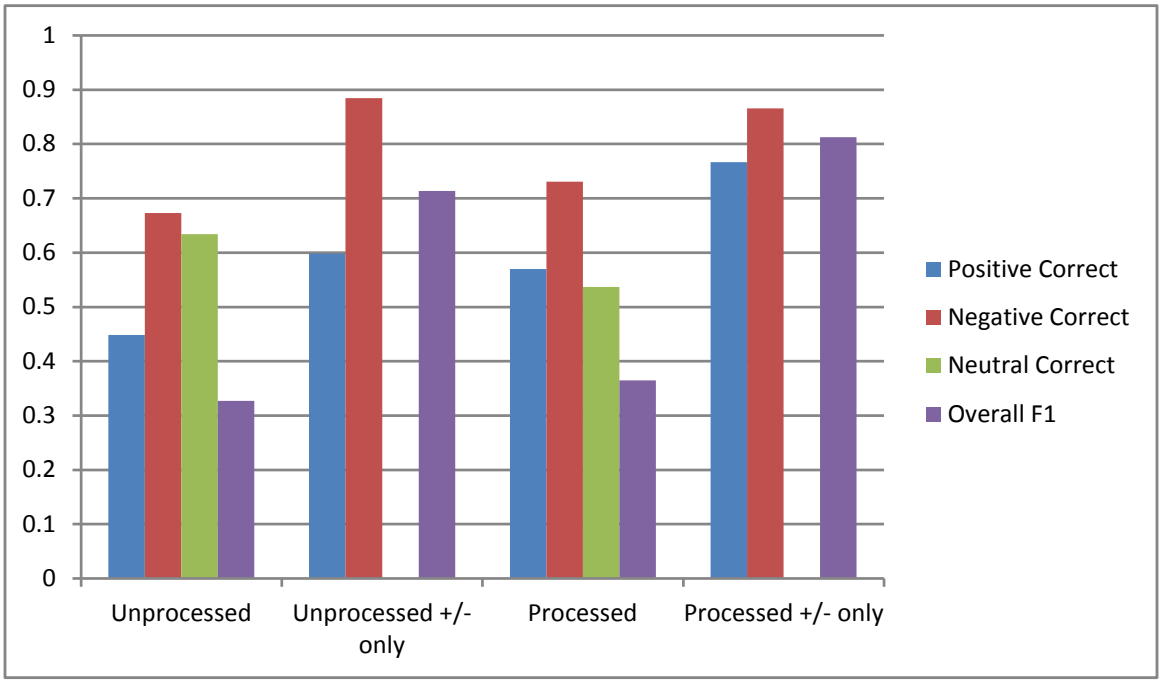
Where n is the length of the review, and $r_i$is the ith word of the review.

The classifier is a robust implementation, but still fails in certain cases discussed below. The classifier was tested on two and three classes, as well as with and without preprocessing of data.

Results

Data Set	Positive Correct	Negative Correct	Neutral Correct	Overall F1
Unprocessed	0.4486	0.6731	0.6341	0.3272
Unprocessed +/- only	0.5981	0.8846	N/A	0.7137
Processed	0.5701	0.7308	0.5366	0.3650

Processed +/- only	0.7664	0.8654	N/A	0.8129



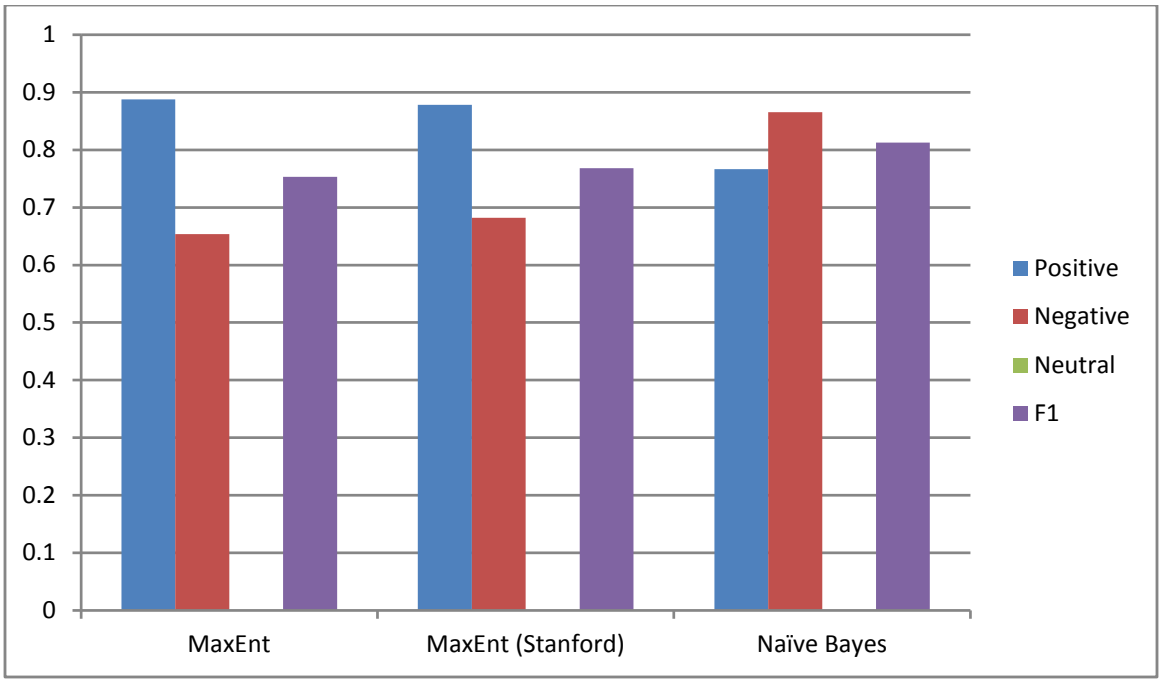
Error Analysis

The first bit of analysis performed was adjusting the δ parameter. An initial setting of 1 classified almost all items to the negative class. Through iterative adjustment, a value of 0.15 was used, which gave a balanced performance and a high F1 score overall.

The Naïve Bayes classifier is excellent at detecting amounts of sentiment, but not as good at classifying between positive and negative. The neutral correct scores are much higher in the general case, relative to the MaxEnt classifier. In reading the errors that the classifier produces, the most common case is when a review with a large amount of sentiment is misclassified to the opposite class. For instance, "This is a great class. Unfortunately, not many people know about it or decide not to take it. There are weekly problem sets, take home midterm and an essay final. The topics covered are interesting and are supplemented with interesting lectures" is misclassified as neutral, when in fact it should be positive. However, since the review does not express some of the more extreme sentiment of other positive reviews, the classifier cannot make the connection.

Comparison of Implemented Classifiers

Classifier	Best Data Set	Positive	Negative	Neutral	F1
MaxEnt	Positive/Negative Only	0.8879	0.6538	N/A	**0.7531**
MaxEnt (Stanford)	Processed, +/- only	0.878	0.682	N/A	**0.768**
Naïve Bayes	Processed +/- only	0.7664	0.8654	N/A	**0.8129**



The MaxEnt classifiers exhibited similar behavior, namely strong performance on positive reviews relative to performance on negative reviews. This is likely due to the fact that because the starter data contained more positive reviews than negative reviews, the classifier was biased toward the positive reviews when it faced reviews with relative uncertainty. On the other hand, the Naïve Bayes achieved similar performance between the two classes. The independence assumption of Naïve Bayes is the main cause here, so as mentioned above, reviews which contain strong sentiment thus become more difficult to classify one way or another, as certain words can indicate sentiment without indicating direction (e.g., very).

Suggestions for Improvement and Future Work

The MaxEnt classifier has great potential to be improved, as the basic feature set gives it good performance, but through the addition of custom features, there could easily be more performance gains seen in this classifier. Based on some of the observed trends, Named Entity Recognition has great potential to improve classification results, by tokenizing grades, professors, and courses into a standard form. This could also provide insight into how certain professors are reviewed, the average grade earned by students who give a particular review, and potentially related classes.

The Naïve Bayes classifier could benefit from a bigram approach. The unigram model performs well, but the independence assumptions cause it to break down in cases that would otherwise be relatively straightforward. The best option here would be to provide additional context to the classifier by upgrading to a bigram or trigram model instead.

Code Notes

Data

Training and testing data are provided in the courserank.train and courserank.test files, respectively. There are also .processed files, which apply the NOT filtering and punctuation removal to the raw data.

Running the System

The system is built using ./ant, as is the case with all prior projects. To run the MaxEnt classifier, simply run

java –cp classes cs224n.assignments.MaximumEntropyClassifierTester <data name>

To run the Naïve Bayes classifier, run

java –cp classes cs224n.assignments.NBClassifierTester <data name>

Both classifiers accept command-line flags "-m yes" to print mistakes, "-n yes" to exclude neutrals, and "-p yes" to run preprocessing steps (NB only).

To run the preprocessing system on a separate file, run

java –cp classes cs224n.assignments.DataPreprocessor <infile> <outfile>

The Stanford classifier is built using make and run via

java –jar stanford-classifier.jar –prop <properties file>

Any data file can be substituted as shown above.

Sources

Li Zhuang, Feng Jing and Xiao-Yan Zhu. "Movie review mining and summarization." 2006. Proceedings of the 15th ACM international conference on Information and knowledge management, pp. 43-50.

Bo Pang, Lillian Lee and Shivakumar Vaithyanathan. "Thumbs Up?: sentiment classification using machine learning techniques." Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10, pp. 79-86.