Statistical Natural Language Processing

Final Project

Sentiment Classification in Twitter: A Comparison between Domain Adaptation and Distant Supervision

Authors:

Michael Khanarian

David Álvarez-Melis

December 20, 2012

Contents

Abstract

In this paper we study empirically the accuracy of various NLP methods to classify Twitter sentiment. We first try the more traditional approach of using labeled movie reviews as a training set and then attempt a less conventional technique of using emoticons in Tweets as noisy labels of the sentiment. We compare the advantages and disadvantages of each approach and determine whether certain modifications to the pre-processing of reviews or Tweets has any significant improvement on accuracy. The methods we implement are Bag of Words, Maximum Entropy, Perceptron, Averaged Perceptron and Bootstrapping our training set. At the end of the paper we provide a summary of the best and worst performing methods as well as discussion of the techniques used.

1 Introduction

Sentiment analysis is an area of enormous potential where machines can take language learning to a higher level and piece together word-level information to build a more comprehensive understanding of overall sentiment. While the techniques of sentiment analysis can vary from very simple to very nuanced, existing research has already established good performance on the task of binary (positive/negative) sentiment classification as well a multiclass (positive/neutral/negative) classification on large bodies of text.

In this paper we take the challenge of sentiment analysis to Twitter data, a more modern domain that presents new challenges. Twitter is a booming phenomena; there are more than 200,000,000 active users who Tweet over 340,000,000 times a day[1]. The sheer magnitude of Twitter therefore would make Tweet sentiment an invaluable source of information for reporters, businesses, politicians and anyone else who wanted a widespread, up-to-date opinion poll on a topic. Despite its size, Tweets remain unapproachable for NLP models because of the lack of labeled data that indicates sentiment, grammatical structure or other basic linguistic features.

1.1 Twitter and Sentiment Analysis:

Using Twitter data for the task of sentiment analysis presents additional challenges as well unique opportunities that must be incorporated. Tweets are informal, unstructured messages lacking many features often found in professionally written text; namely, sound grammatical structure and use of a standard lexicon. Furthermore, Tweets are limited in length by 140 characters making them exceptionally brief and therefore the message of the user is very compressed and often times contextual. These challenges, however, are balanced by certain unique features about Tweets that most standard text omit, specifically the use of emoticons and hashtags.

What is special about Tweets including emoticons is that the sentiment is often explicitly specified in the Tweet itself by the emoticon, making it trivial to classify. Therefore, we can use this feature to overcome one of the largest obstacles of analyzing Twitter sentiment, the lack of a sufficiently large labeled database for training. By using this method of labeling

[1]FORBES, June 2012.

Tweets based on their emoticons we can query Twitter and create a large data base of noisily labeled Tweets. Details of this method of learning are explained in subsequent sections as well as how hashtags and other features unique to Twitter were incorporated.

Because the noisy label approach to analyzing Tweets is highly experimental, we also used a more rudimentary approach to the problem by training on a large corpora of IMBD movie reviews with an accompanying numerical score. The motivation for using movie reviews was that texts cover a wide range of topics and contain modern references similar to those in Tweets, such as famous actors and places.

These two approaches are applied to our queried Tweets and their respective performance is compared. We also present additional techniques and considerations such has dividing the data into binary and ternary labels as well as generating a larger training corpus using a bootstrapping method.

2 Previous Work

Sentiment analysis is a relatively new field within natural language processing and data mining, whose foundations were set in a seminal paper by Pang et al. (2002). Various approaches to the problem have been proposed over the last decade (see for example the survey on Pang and Lee, 2008). The advent of social media has also had an impact on the interest in classifying sentiments on shared opinions and

Twitter, and microblogging in general, is a more recent phenomenon, and thus there has been less work done on this context. This, however, is rapidly changing, and as Twitter increases in popularity so does the academic interest in it (Read, 2005; Yang et al., 2007).

One of the first papers to deal specifically with tweets as a framework for sentiment analysis is Go et al (2009). Here, the authors used a distant learning approach through emoticons and trained classifiers using SVM, Naive Bayes and Maximum Entropy, on unigram and bigram features. Training and testing only on positive and negative classes, they obtain an 83.0% accuracy on a set of 359 hand-labeled tweets, but perform badly on three-category classification. Their best result corresponds to using a Maxent classifier with both unigrams and bigrams, although Naive-Bayes has a better average performance over different feature

scenarios. They also conclude that adding POS tags has little to no positive effect on accuracy.

Pak and Paroubek (2010) build upon this and use POS tags to distinguish objective and subjective parts-of-speech, in addition to using only n-grams with high salience and/or low entropy. The use only a Naive-Bayes classifier, and provide a wide range of results, as a function of varying *decision* and *accuracy*.

In a different approach, Davidov et al. (2010) adopt a k-nearest neighbor (kNN) strategy on specific features and train on a database of 475 million tweets. They also use a set of 50 hashtags hand-labeled as positive or negative and then classify tweets based on this. The range of experiments carried out is larger than in the previous papers. Here, the authors try multi-class classifications on smileys and hashtags, and binary neutral/non-neutral classifications.

In this work, we use a similar approach to (Go et al, 2010) of using emoticons as noisy labels, although we also add hashtags as binary labels (as opposed to multi-class category labels in Davidov et al, 2010). We also use an approach similar to (Pak and Paroubek, 2010) to create a neutral sentiment datatset.

3 Approach

3.1 Domain Adaptation

The need for domain adaptation is motivated by the difficulty in generating labeled data for NLP tasks on specific types of text. For some domains this is not a problem because there already exist corpora that are sufficiently similar in distribution to the test case, and training on a body of text such as the Penn Treebank is sufficient. In fact, there are a number of theoretical results in existing machine learning literature regarding test error bounds when the training and test samples come from the same distribution of data and the sample sizes are sufficiently large. Unfortunately for our task, finding a large corpora of labeled Tweets is not possible and hence alternative measures must be applied. The two main qualities desirable for our ideal labeled data would be a certain brevity in each unit of text as well as a highly informal level of writing. While the latter is difficult to achieve, we focus on applying domain adaptation to get closer to the first characteristic.

The concept of domain adaptation is that a large labeled corpus from a different domain

can still be of use for testing on a different domain if the training corpus is sufficiently large and varied so that it would capture some of the features present in the testing set. Furthermore, we can make various adjustment that when applied to the text, preserve its value as a labeled corpus while bringing it closer in distribution to the domain of the test data.

One barrier between the movie review domain and the Tweets is that there is a significant difference in length of the text; on average 600 words for each review vs. 140 characters maximum for Tweets. Consequently, a method that can preserve the sentiment of the review while stripping out the majority of the text would be desirable. One way to achieve this is to make the assumption that movie reviews, like most articles, summarize their opinion in the beginning and final sentences and most of the body text is elaborating on the introduction and building towards the conclusion. Hence it is fair to assume that shortening the reviews to include only these first two and last two sentences may be an effective way to capture a more compressed version of the reviewer's sentiment.

While the above method handles the length difference between our training and test sets, capturing the unstructured nature of Tweets is a much more difficult task that did not have any immediate remedy. One potential source of more labeled, unstructured text could be reviews written by internet reviewers rather than professional movie critics. Our hesitation at this technique was determining a meaningful manner to combine and correspond the restaurant/product reviews with movie reviews on the same scale.

3.2 Distant Supervision

Distant supervision is a learning technique that makes use of a "weakly" labeled training set, where labels are considered to be "weak" or "noisy" whene obtained based on a heuristic function or on side information. Thus, these labels have no guarantee of providing an accurate tag. The sole existence of these noisy labels sets Distant Supervision within the framework of semi-supervised learning, and

Although other authors had previously explored ideas resembling noisy labels before (particularly in bioinformatics, such as in Craven and Kumlien, 1999), it was until Mintz et al (2009) coined the term *Distant Supervision* that the concept was formally established.

Emoticons where first proposed as noisy labels for sentiment classification by Read (2005)

and later used -branded now as distant supervision- in the particular context of Twitter by Go et al. (2009). As the former explains, emoticons have been developed in electronic means of communication as visual cues associated with emotional states, particularly with that of the author of text accompanying these glyphs. Thus, a simile (see Table (ref) for examples) can assumed to convey a positive feeling, while a sad face generally conveys a negative feeling.

Using a distant supervision approach for this problem provides many advantages. First, it tackles the problem of scarcity of labeled data: by querying Twitter for entries containing "happy" or "sad" emoticons, there is virtually[2] no limit to the size of the training set, which, in addition, is completely free to obtain. This process is also relatively fast, as we explain in section (ref), we were able to build a database with almost 4 Million entries in a matter of days. Finally, and most importantly, this approach can be generalized by adding other possible noisy labels from within Twitter-specific features, such as certain hastags (#) or user mentions (@).

An issue that might pass unnoticed at first about using internal features of examples as noisy labels, but turns out to have major repercussions, is the handling of these labels during training. The problem arises because training instances that contain the feature that labels them will almost always cause any method to only learn from those features and not from other characteristics of the examples. In our context, it would cause out method to only learn emoticons and no linguistic features of the Tweets. Both Read (2005) and Go et al (2009) tackle this issue by stripping emoticons before training, which nevertheless, has the disadvantage of preventing *learning* from emoticons, and thus causes the method to ignore a very valuable source of polarity. To circumvent this limitation, we tried a bootstrapping approach of adding highly ranked examples (with or without emoticons) of an additional held-out data set into the training set, and then retraining on this set. This procedure is explained fully in the Section 5.5.

[2]There is in fact a limit on the amount of Tweets and can be retrieved from Twitter's server; 150 queries an hour per IP address. However, by setting up scheduled queries without exceeding these limits, it is possible to have a continuous, limitless source of labeled data.

4 Corpus Collection and Processing

4.1 Movie Reviews

As discussed earlier, we decided to use movie reviews for the purpose of training our models in the hope that a sufficiently large labeled corpus could be used to test on a different domain with modest results. The movie reviews we used are recent IMBD reviews freely available through the Cornell Movie Review data site. This site contains a series of datasets containing large bodies of text with a corresponding score of sentiment or polarity.

The set we used specifically was a combination of 5000 reviews written by four professional movie critics who then scored the movie using either a 4-star or 5-star scale. The star scores were already rescaled onto the $0.0 - 1.0$ interval in increments of 0.1 Pang et al. (2002). We intentionally selected the dataset with the most finely scaled ratings because we could divide the reviews into binary of ternary classes at our discretion.

Additionally, the reviews only required mild pre-processing involving the removal of foreign language words and reviews that started with unrelated information about the movie. The more interesting part was classifying the numerical scores into two classes (positive/negative) as well as three classes (positive/neutral/negative) in a consistent manner.

For the two class problem we decided that the empirical mean of the ratings on the training data (0.588) was a fair boundary between the positive and negative classes. We considered using the median score as well to balance the number of positive and negative training reviews but found that the dividing boundary nearly coincided with that of the mean.

The two class boundary is flawed, however, because many of the reviews that are close to the boundary are not all that different from each other and really belong in a category of neutral sentiment rather than forced into either a strictly positive or negative category. Creating a third category for the reviews corresponding to ratings near the middle is a less obvious a task. What we decided upon was to use the standard deviation of the ratings (0.18) and deem that all reviews corresponding to ratings within +/- 1 standard deviation of the empirical mean $(0.4 - 0.76)$ to be in this third neutral category. The idea of distinguishing the middle class by the first standard deviation seems to coincide approximately with the bounds in Go et al. (2009).

4.2 Tweets

As briefly discussed in section 3.2, under the framework of noisy labels datasets are obtained by sending queries to Twitter's website for positive or negative emoticons. In a first instance, we used the same database used in Go et al. (2009), which we will denote as GO-Train, in order to replicate their results. The tweets in this set are already processed (they have had emoticons stripped) and includes only negative and positive tags. There are a total of $800,000$ positive tweets and the same amount of negative tweets.

Then, we set upon the task of creating our own database, with the following additions to Go's. First, we added a third query, aimed at obtaining neutral-sentiment examples, by collecting tweets from 10 of the most popular newspaper accounts (similar to Pak and Paroubek, 2010) on Twitter according to Forbes[3]. This list includes the New York Time (@nytimes), the Wall Street Journal (@wsj) and the Washington Post (@washingtonpost) among others. Secondly, we added some very frequently used hastags to the positive and negative query set, which we considered could also act as noisy polarity labels, along with other *smileys*. The full set of glyphs used is shown in Table

Table 1: Emoticons and handles used in Twitter queries

Sentiment	Queries
Positive	:-) :) :) =) :D ;D ;-) <3 #happy #rocks #iloveit
Negative	:-(:(: (=(;(#sad #sucks #bored
Neutral	from:nytimes, from:coloneltribune, from:wsj, from:washingtonpost, from:latimes, from:ajc, from:freep from:usatoday, from:newyorkpost, from:denverpost

According to Twitter's Terms of Service, the amount of queries by a single user is limited to 150 per hour. Thus, we implemented a `python` routine that sent a query every 28 seconds, and stored the data as `.atom` files. This time lapse also allowed for the Tweet list to be refreshed, in order to avoid duplicated elements. We collected 3.9 million tweets, distributed evenly over the three sentiment categories. After all the files were collected, a `.csv` file collected all the Tweets, which was the processed as follows.

(i) All query terms in Table 5 were removed for the training set. For the heldout (boost-

[3]Full list available at http://www.forbes.com/sites/jasonoberholtzer/2011/04/12/the-top-25-newspapers-on-twitter/.

ing) and test sets, they were replaced by the tokens *HAPPY_TAG*, *SAD_TAG* and *NEUTRAL_TAG*.

(ii) Tweets containing conflicting noisy labels (i.e. at least one from more than one category in Table5 (which might happen if there are different sentiments being attached to various ideas in one entry) were removed.

(ii) User mentions (@) and URL addresses were replaced by the tokens *USERMENTION* and *URL*, while hashtagged words (e.g. #(somestring)) were replaced by post-fixed tokens (somestring)*_HASH*, in order to capture the effect on sentiment using the hashtag version of a word, instead of the word alone.

(iii) Abbreviated expressions such containing *'d* and *'t* were reverted to their complete form *NOUN + would* and *VERB + not*.

(iv) Repeated and Re-tweeted entries were removed, to avoid giving excessive importance to particular tweets.

After these modifications, the size of the dataset shrank to $3,330,077$ entries. Thus, the final sizes of the training and held-out sets was $2,640,061$ and $690,016$, respectively. We denote these in the following as OUR-Train and OUR-Held.

4.3 Testing Set

Since the original task of this work was to perform sentiment analysis on Twitter, we used datasets of tweets for testing both approaches explained in Section 3.

To this end, we used two different test sets. The first one is Go's test set (denoted GO-Test), consisting of 457 hand-labeled tweets, of which 182 are tagged as positive, 177 as negative and 139 as neutral. Since this set was deemed to be too small, particularly when compared to the large sizes of the training sets in play, a second -larger- database was obtained. Niek Sanders' website[4] provides a database with 5513 labeled tweets, of which 570 are positive, 2503 neutral, 1187 negative, and the remaining 1786 are considered to be irrelevant (spam and non-english language tweets). We removed this last category, and thus were left with $4,260$ entries in the database, which we denote as SANDERS.

[4]http://www.sananalytics.com

Both of these sets are pre-processed before being used by our method, with the same criteria described above for the training set.

5 Methods

In this section we present the methods used to implement the Twitter sentiment classifier under both approaches, Domain Adaptation and Distant Learning. All methods, code, secondary programs and IO scripts were coded by the authors. Some of it, however, was taken and adapted from previous implementations used for previous assignments in this class.

5.1 Bag-of-Words

To establish a baseline for our models we applied a simple "bag of words" approach for both training on movie reviews and Tweets. This approach is very straightforward and we simply hand select words that we feel indicate either positive or negative sentiment in a general sense. When a test sample is encountered, the model simply scores the text based on the occurrences of positive words and negative words and assigns a label based on which category scores higher.

To provide more detailed information on the effectiveness of this "bag of words" method we first used a "mini bag" of only three keyword for each label category (Table 2) and then expanded to including 20 words for each label. The expanded set of words used for movie reviews is shown in Table 4:

Table 2: Mini Bag of Words for Movie Reviews

Sentiment	Words
Positive	best, great, awesome
Negative	worst, bad, awful

For Twitter we modified the bag of words to reflect the more casual nature of the text. Therefore our "mini bag" included the following words:

The accuracy improvement from this very simple Mini Bag was quite astonishing, especially for the Tweets, as we demonstrate in the results section. To have a more reliable baseline, however, we included a more comprehensive bag of words which is listed below.

Table 3: Mini Bag of Words for Tweets

Sentiment	Words
Positive	good, rocks, love
Negative	bad, hate, sucks

Table 4: Full Bag of Words for Movies

Sentiment	Words
Positive	Mini Bag + terrific, good, funny, unique, encouraging, eloquent etc...
Negative	Mini Bag + weak, awful, terrible, negative, lame, boring, dull etc...

5.2 Maximum Entropy

After establishing our baseline with the bag of words approach we decided to use a more statistical approach to extract more information from the training data. Therefore we modified the Maximum Entropy program used in earlier assignment to extract N-grams as features from our training reviews and then use this information for the task of classifying sentiment.

The general idea is to generate "scores" for each label and convert each to a respective probability via:

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{e^{w^T f(y)}}{\sum_{y'} e^{w^T f(y')}}$$

5.3 Perceptron

We used a multiclass perceptron classifier, which had been coded for a previous assignment and had yielded good results. Only minor modifications were necessary to adapt it to this context.

As usual, the perceptron updates the weight vector w_t in an online fashion, by obtaining

$$\hat{s} = \arg\max_{s} f(t, s) \cdot w$$

where f is the feature representation of the tweet t given its sentiment s, and $\hat{s}$ is the sentiment class predicted by the method. Then, only when the predicted sentiment does not match the actual sentiment s^* at step τ, the weight vector is updated as follows:

Table 5: Full Bag of Words for Tweets

Sentiment	Words
Positive	Mini Bag + sexy, hot, free, best, amazing lol, happy etc...
Negative	Mini Bag + f*ck, b*tch, garbage, broken, kill, waste etc...

$$w_{\tau+1} = w_\tau + f(t, s) - f(t, \hat{s})$$

In our implementations, we first randomize the order in which training examples appear (to avoid leaning towards a certain class at the end of training) and we revisit the whole corpus a total of *it* times, where *it* is a parameter to be determined on a case-by-case basis.

5.4 Averaged Perceptron

Given good results in previous assignments, we decided to also implement a slight modification to the perceptron detailed in the previous section, where the final wight vector is an average of the weight vectors produced after each round of training over the data. Thus, the optimal wight vector is chosen as

$$w^* = \frac{1}{\text{it}} \sum_{i=0}^{\text{it}} w_i$$

where, again, *it* is the number of iterations over the training data.

5.5 Bootstrapping

In order to deal with a the problem of training with or without the emoticons in the database, we tried the following bootstrapping approach. We split the main database into a training set $\mathcal{T}$ and a held-out set $\mathcal{H}$, in proportions 8-to-2. Only in the former were the emoticons stripped, in the latter they were kept but only replaced for a general token *SAD_EMOTICON* or *HAPPY_EMOTICON* depending on their type.

We first trained the method on $\mathcal{T}$ and then obtained the confidence values for the set $\mathcal{H}$. From there, we took all Tweets for which the method yielded a confidence level above $\rho \in (0, 1)$ and appended them to $\mathcal{T}$, creating a new training set $\mathcal{T}^*$. Then, the method was retrained on this definitive set.

The advantage of this approach is that, as opposed to (Go et al., 2009), our method does

learn polarity from emoticons, and in fact, predicts very accurately when one is found in a test example. However, by only adding those Tweets for which the method had originally assigned a high confidence level, we are preventing the algorithm to over-learn on these while neglecting linguistic features.

In Section 6, we provide results for both using and not using the bootstrapping approach.

6 Results

6.1 Domain Adaptation

The following results are for standard binary classification as well as binary classification with the neutral category removed per our discussion earlier. When using the Maximum Entropy program from our earlier assignment we performed 100 iterations to generate the results below:

Table 6: Classifier Accuracy on Reviews

Model	Pos/Neg	Pos/Neg no Neutral
MiniBag	61.6	62.4
Bag of Words	65.6	65.5
Uni	72.6	86.6
Bi	79.1	86.8
Uni/Bi	77.2	88.8

What we observe is that our Bag of Words model performs surprisingly well indicating that our model can perform better than random by querying the test instance for only a handful of words. The statistical approaches do perform considerably better, however, and it seems extracting bigrams outperforms all. While these numbers are encouraging, it is improbable that comparable level of accuracy can be maintained when testing on a different domain as we see in the subsequent section. Knowing that we must adjust our database to accommodate the different nature of the Twitter data we know implement our idea of compressing the review to the beginning two and end two sentences.

When we then applied our domain adaptation technique of compressing the reviews this performed better than when training on the entire review as seen below, however the improvement was not nearly as significant as we had hoped. While compressing the reviews via

Table 7: Accuracy on Compressed Reviews/Testing on Go Database

Model	Pos/Neg
Uni	61.3
Bi	62.4
Uni/Bi	62.3

topic sentence is a logical idea, these abbreviated reviews still did not capture the brevity exemplified by Tweets which explains why the domain adapted training performs so poorly.

6.2 Distant Learning

Training and testing on the same database as Go et al. (2009), we obtained the following results. Unless otherwise specified, Maxent methods had a maximum iteration criterion of `maxit`=100, and both the Perceptron and Averaged Perceptron were iterated 20 times over the training set. In the rightmost column we include the results of Go's Maxent classifier for comparison.

Table 8: Classifier Accuracy trainig/testing on Go Database

Features	Baseline	IMDB Maxent	Twitter Percep	Twitter AvPercep	Twitter Maxent	Go et al. Maxent
Unigrams	62.1	55.5	78.7	80.5	79.3	80.5
Bigrams	–	58.1	77.2	79.8	80.1	79.1
Uni/Bi	–	57.2	78.9	80.2	82.4	83.0
Uni/Bi+Feat	–	–	78.8	78.49	79.8	–

The results on Table 8 show that our maxent implementation closely resembles theirs, for we where able to obtain similar results overall. Our additional features not only did not improve accuracy, but actually brought it down both for the averaged perceptron and the maxent classifier. However, when tested on the second test set, performance was much worse, as seen in Table 9. For this point onward we no longer present results for the worst performing methods, that is, IMDB Maxent and Twitter Perceptron.

As we can see, the classifier does not seem to generalize very well to other (larger) test sets. Since our results on Go's database are similar to theirs and our implementations are symmetric, we suppose that their method would also suffer from the same problem.

Now, we train the same models on our database, and try them out on both test sets for

Table 9: Classifier Accuracy training on Go Database, testing on Sanders

Features	Twitter AvPercep	Twitter Maxent
Unigrams	61.81	63.17
Uni/Bi	62.55	64.55

comparison. When using bootstrapping, we take the confidence threshold as $\rho = 0.9$ (see previous section), which we found to provide the best results on average. Also, to make the method more efficient, we reduced maximum iteration criteria for the first round of training, by setting `maxit`= 30 for maxent. Table 11 summarizes the results obtained for several configurations.

Table 10: Classifier Accuracy training on OUR, testing on GO/SANDERS.

Model	Bootstrapping	Acc. on GO	Acc. on SANDERS
Averaged Perceptron	No	73.01	70.16
Averaged Perceptron	Yes	76.75	72.89
Maxent	No	78.27	71.51
Maxent	Yes	79.12	73.56

As we can see, training on our database has significantly improved the performance on SANDERS, but has damaged the performance on GO. This puzzling effect would seem to suggest that the test set used by Go et al. is better correlated to their training set. This is certainly a curious coincidence, that could nonetheless be explained by the fact that both were obtained in the same time period, and therefore could include Tweets from the same users, or capture sentiment on similar topics. On another note, it is interesting to note that in all cases the use of bootstrapping increased the accuracy. Also, it is remarkable that the averaged perceptron obtained very similar results to the maxent, considering how simpler and faster it is than the latter. Also, it offers the advantage of being an on-line method, and thus for a real-time implementation, an averaged perceptron classifier could be an efficient option.

Finally, we tried our methods for multiclass classification, that is, adding a NEUTRAL label into this analysis. Since GO-Train does not have neutral tweets, the following experiments could only be done in our database, and were assessed, again, both in GO-Test and SANDERS.

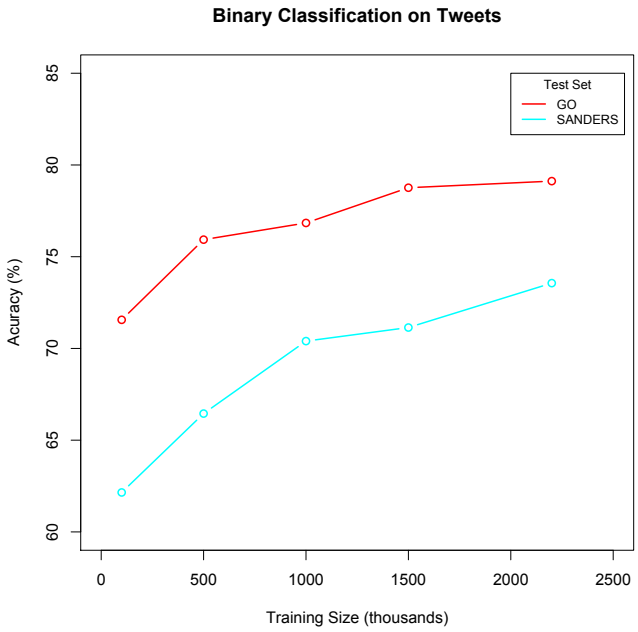

Figure 1: Accuracy as a function of training set size (in thousands).

Table 11: Classifier Accuracy training on OUR, testing on GO/SANDERS.

Model	Bootstrapping	Acc on GO	Acc on SANDERS
Averaged Perceptron	No	53.83	27.76
Averaged Perceptron	Yes	57.45	28.45
Maxent	No	58.87	38.83
Maxent	Yes	60.68	39.43

The dissimilar values in columns three and four come from the fact that SANDERS has a significantly larger proportion of neutral Tweets (almost two thirds) compared to GO (one third). This would suggest that our method is still not very good at identifying neutral tweets, and a more advanced approach should be adopted to deal with them. However, for the best performing method, the Maxent with bootstrapping, the results are certainly not bad and show promising foundations for a Pos-Neg-Neutral classifier.

6.3 Error Analysis

We now analyze some of the errors made by our best performing method, the maximum entropy classifier with bootstrapping. Some of these errors are obvious and denote clear deficiencies in our approach. For example

Error: Cheney and Bush are the real culprits - LINK guess=POS gold=NEG confidence=0.9999999999999662

Here, although the tweet is short, the presence of the word "culprit" should have given enough information to classify the sentiment as negative, although the method does the opposite. This could happen because "culprit", being a sophisticated and formal (at least for Twitter standards) word, was not seen before in the training set, and thus the method had no elements to discriminate based upon it.

However, there are many cases in which the sentiment of the tweet is much more subtle and is embedded in the semantics and not in the grammar. Let us consider, for example, the following mis-classified entry.

Error: LINK With friends like HASH_Billgates and HASH_steveballmet at HASH_microsoft, who needs enemies HAPPY_EMOT guess=POS gold=NEG confidence=1.0

The method predicted a positive label, which considering the occurrence of the word "friends" and the presence of a positive emoticon, seems like a sensible choice. However, the poster is actually conveying disapproval about the subjects of this message, Bill Gates and Steve Ballmet, through the use of a well known ironic proverb. It is clear that these type of mistakes are much harder to avoid, and require a more sophisticated approach.

7 Conclusion

We have approached the problem of classifying the sentiment of a Tweet from two very different perspectives; through Domain Adaptation with movie reviews and through distant learning, using emoticons as noisy labels. Different methods under the two frameworks were implemented, and their performance was evaluated on a test set of hand-labeled tweets.

The results clearly favor distant learning over domain adaptation. This, however, is not surprising, and it was a feasible scenario from the beginning. The reason is clear; the domains of movie reviews and tweets are completely different, so much as to make almost impossible any generalization from one to the other. Let alone the format, type of language and profiles of authors of both kids of texts, there are two other crucial differences between them. First, reviews are made with the sole purpose of conveying an opinion on a specific topic, and thus

are meticulous in justifying the sentiment conveyed. Tweets, on the other hand are not, and when they happen to carry a sentiment it does not have to be argued for or justified. This evidently has an impact on the choice of words and type of sentences formulated in each case.

The second big difference is a consequence of the size of both types of texts. The limit of 140 characters in tweets seldom allows one to convey a developed opinion on multiple topics, while this is very common in reviews, where the author might praise some aspects of a movie but criticize some others. This also creates a barrier on any method that "learns" sentiment on one source and tries to apply it to the other. By compressing the reviews into their topic and concluding sentences we were able to see some improvement in accuracy however these abbreviated reviews still lacked the brevity and exclamatory nature of the Tweets.

In any case, the distant learning approach with noisy labels proved to be a powerful method, which can generate training sets of unlimited size and thus greatly improve accuracy levels. Even though our method train on the improved database did not achieve the same performance as the one trained on the external database, the results suggest that it might generalize better to other test sets. This should be explored further, by testing on different, preferably larger labeled databases.

Other possible improvements could come from a more careful handling of emoticons and hashtags in training instances, maybe creating classes instead of grouping into a single token. Davidov et al. (2010) offer some insights into this. There is also a lot of work to do in terms of dealing with neutral tweets.

In terms of improving the domain adaptation method, our goal has to be towards finding labeled data that captures the brevity seen in Tweets. One possible solution could be to use magazine/tabloid headlines and story titles that reflect sentiment in a similar brief fashion to Tweets.

References

[1] http://www.cs.cornell.edu/people/pabo/movie-review-data/.

[2] D. Davidov, O. Tsur, and A. Rappoport, *Enhanced sentiment learning using twitter hashtags and smileys*, in Proceedings of the 23rd International Conference on Computational Linguistics: Posters, COLING '10, Association for Computational Linguistics, 2010, pp. 241–249.

[3] G. et al., *Part-of-speech tagging for twitter: Annotation, features and experiments*, in Proceeding of the 49th Annual Meeting of the ACL: Human Language Technologies, vol. 2, Association for Computational Linguistics, pp. 42–47.

[4] A. Go, R. Bhayani, and L. Huang, *Twitter sentiment classification using distant supervision*, Processing, (2009), pp. 1–6.

[5] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, *Distant supervision for relation extraction without labeled data*, in Proceedings of ACL-IJCNLP 2009, 2009.

[6] A. Pak and P. Paroubek, *Twitter as a corpus for sentiment analysis and opinion mining*, in Proceedings of the Seventh International Conference on Language Resources, European Language Resources Association, 2010.

[7] B. Pang, L. Lee, and S. Vaithyanathan, *Thumbs up?: Sentiment classification using machine learning techniques*, in Proceeding of the ACL-02 conference on Empirical Methods in Natural Language Processing, vol. 10, Association for Computational Linguistics, 2002, pp. 79–86.

[8] J. Read, *Using emoticons to reduce dependency in machine learning techniques for sentmient classification*, in Proceedings of the 43rd Meeting of the ACL, Association for Computational Linguistics, 2005, pp. 43–48.