9. Extracting Weather and Sentiment from Tweets

December 11, 2013

1 Introduction

There are approximately 200 million active users on Twitter and 400 million tweets are posted on a daily basis. With such a large volume of tweets there is huge potential for gathering information about public sentiment. We would like to know whether our machine learning tools could viably be used to leverage this data and predict public opinion. The specific domain of weather is particularly rich weather is a common topic of conversation, and this is no different in the Twitter sphere. For example, people may use twitter to express their joy about a beautiful sunny day or to complain about the consistent rain. Therefore, we aim to build a classifier that can extract a tweet's sentiment, time frame, and weather condition by tailoring our off-the-shelf learning algorithms to this unique setting. Then we can examine our different classifier to determine which algorithms are most effective.

To build our classifier we used three different algorithms: decision trees, support vector machines (SVM) and Naive Bayes. Each algorithm was modified to fit our problem and model selection was used to determine the best-input parameters. Our findings showed that each of the algorithms had its pros and cons with better performance in different aspects of the problem. In addition, we attempted to build a classifier using a Markov Random Field, which would be an improvement on the Naive Bayes algorithm that factors in a tweets location and time of posting. In this report we will further discuss the methods used and provide a detailed analysis of the experimental results.

2 Problem Definition and Methods

2.1 Task Definition

Through our project, which was one of the Kaggle Projects assigned, we aimed to calculate classification probabilities for incoming tweets into 24 labels. These 24 labels came from 3 main categories:

- Sentiment = {Cant tell, Positive, Neutral, Negative, Unrelated to weather}
- When = {Current, Future, Cant tell, Past}
- Kind = {Clouds, Cold, Dry, Hot, Humid, Hurricane, Cant tell, Ice, Other, Rain, Snow, Storms, Sun, Tornado, Wind}

Our task was to take an incoming tweet and classify it to produce a vector of 24 confidence scores: one for each label. This confidence score indicated a level of confidence for that particular tweet to have significant information about that particular label. The categories of Sentiment and When could only have one classification with a positive confidence score (as for instance, a weather related event cannot happen in the past as well as in the future), but there could be multiple positive confidence scores for labels in the Kind category (it can be “cloudy” as well as “rainy” at the same time). Our training set contained tweets, locations, and a confidence score for each of the 24 possible labels.

For each tweet, we would like to have definitive “sentiment”, “when”, and “kind” values

2.2 Algorithm

Given our input training sets (Kaggle and Cheng-Caverlee-Lee) and constraints, we decided to use a set of algorithms that we felt would best classify our data. The categories of Sentiment and When were discriminative in nature and therefore merited the use of discriminative algorithms, whereas the Kind category was generative (requiring a real-valued number representing a “score” of how likely that kind of weather was, given the tweet. A single tweet could thus have multiple kinds of weather at once. Thus the algorithms we ended up using were:

1. Support Vector Machine

   We converted our input dataset to the Svmlight format and used Professor Joachimss Svmlight module to learn and classify each of the 24 separate labels as binary classifiers. We then took the max of the resulting margins within the categories to single out the best labels.

2. Decision Tree

   We used the TDIDT ID3 algorithm to build Decision Tree classifiers for each of the 24 labels, with splitting criteria ≥ 1 and ≥ 0. We then took the max of the resulting classifications to similarly decide on the best labels.

3. Nave Bayes

   We used independent class conditional probabilities of each significant word (or combination of words) in the tweet, and used that to calculate the probability of a label being assigned to a tweet.

4. Markov Random Field

   As an effort to make Naive Bayes less “naive”, we introduced conditional dependencies (based on the location and timestamp of a tweet) to classify incoming test tweets better. This was based on the assumption that tweets in nearby locations and within the same intervals of time, would have some level of similarity when it comes to the kind of weather they would be talking about. Therefore, our MRF classifier made the use of these dependencies to classify incoming tweets based on previous tweets, weighing “similar” tweets greater than “dissimilar” ones when making a classification.

Each of these algorithms, along with the constraints we put on them, are explained in more detail further sections.

3 Methodology

3.1 Data Manipulation

3.1.1 Preprocessing

The provided Kaggle data comes with a guarantee that the tweets are either about weather or are labelled in a way that indicates that they aren’t about weather. So, there was no need to filter out entired tweets. But, as expected, a lot of the content of tweets aren't particularly important to our analysis. Stopwords like "a" and "the" are not worth considering in our algorithms. Using a predefined list of stopwords (reference mentioned in Appendix), we filtered out all stopwords from tweets. Moreover, we filtered out some common Twitter-specific jargon that bears no meaning in our problem, e.g. "RT" and "@mention".

Although capitalization may be indicative of strong emotions (and therefore helpful to sentiment), we've chosen to focus on the words themselves, so all words in each tweet are converted to lowercase. Lastly, all punctuation symbols within a tweet are removed. Well, not quite *all* punctuation...emoticons are comprised of punctuation marks and can be quite useful with regards to understanding sentiment, so all emoticons are kept in the tweets and treated as any other word.

3.1.2 Important Words

The discriminative algorithms we use (SVM and decision trees) rely on translating our tweets into feature vectors, where each feature is a word of the vocabulary. Our vocabulary V was very large (on the order of 10^7), so it would be unwieldy to represent each tweet with a vector of length $|V|$, especially since some words aren't as important as others in determining weather/sentiment from tweets. To deal with this issue, we used class conditional probabilities from Naive Bayes (algorithm discussed later) to determine which words are most important in determining sentiment, when, and kind – the notion of "most important" has to do with which words have a high probability of occuring given that, for example, the kind of weather is "rainy".

The most important words for sentiment, when, and kind generated from this analysis are depicted in the below word clouds (larger words are more important than smaller ones). As we can see, most of these important words are indeed related to weather. For instance, it is unsurprising that the word "good" has high importance for sentiment, and "storm" has high importance for kind.

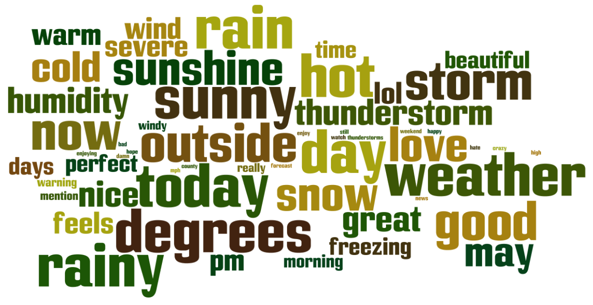

Figure 1: Top 30 "sentiment" words

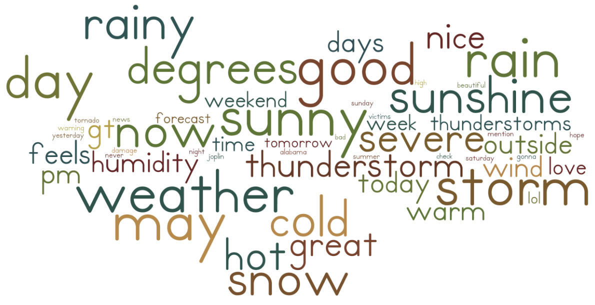

Figure 2: Top 30 "when" words

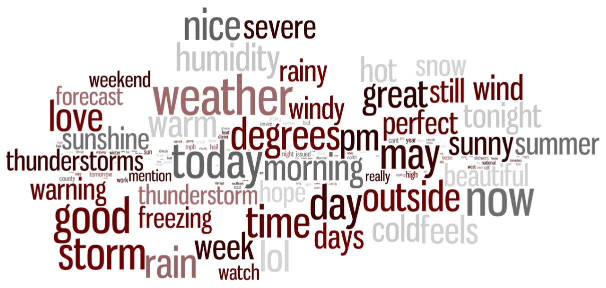

Figure 3: Top 50 "kind" words

3.1.3 Data for MRF

We explore a method for tackling this problem using Markov Random Fields (MRF), which relies on location data and time data. The available Kaggle data doesn't have timestamps, so we utilized a separate dataset (referenced in Appendix). This data set is unlabelled and not weather-specific. To distill out the tweets related to weather, we removed all tweets that do not contain any words from the list of most important words for determining the kind of weather (gathering of important words discussed in previous subsection). The tweets were then labelled using decision trees (algorithm methodology discussed later), which did the best job of determining the kind of weather. Finally, we sent the remaining tweets through the same preprocessing specified in the "Preprocessing" subsection above.

3.2 Preliminary Methods

3.2.1 Decision Trees

Another discriminative algorithm we used to classify the tweets is decision trees. From the homework set, we saw that decision trees could be useful for classifying texts into different genres. Therefore, we used the TDIT algorithm along with information gain to the build decision trees for classifying the "when", "sentiment" and "kind" labels of tweets. Since a tweet can have multiple "kind" labels, we constructed a tree for each of the fifteen labels indicating either "positive" or "negative." Then the vector of "kind" labels can be generated from the output of each of the fifteen trees.

The feature vectors used to represent the tweets were the bag of "important" words constructed from the Nave Bayes conditional probabilities. Thus, each attribute represented a word count. Then the splitting criteria used was either > 0 or > 1 depending on which gave more information gain. In order to prevent overfitting early stopping was used. Lastly, cross validation was used to determine the optimal tree depth from a range of 2 to 60.

3.2.2 SVM

3.2.3 Naive Bayes

We use Naive Bayes to learn the "sentiment" and "when" labels of a tweet, and to learn which kinds of weather are occuring. In addition to implementing the off-the-shelf version of the algorithm learned in class, we also took steps to make Naive Bayes a little less naive.

Traditionally, in this setting, Naive Bayes breaks a tweet up into words and maintains probabilities corresponding to each word. Our implementation of the algorithm allows support for maintaining probabilities corresponding to groups of k words for any k. For instance, if the tweet is "not sunny today :(", we add the pairs of words "not sunny", "sunny today", and "today :(" to our vocabulary (i.e. we look at bigrams). The groups of words consist of consecutive words as in this example. We experimented with different blends of unigrams, bigrams, and trigrams when performing Naive Bayes.

3.3 Advanced Methods

Although the words within a tweet can be strong indicators of the weather conditions surrounding a tweet, it is also helpful to know the location and time of the tweet. For example, if the tweet is made in Boston in December, there is likely to be snow. The traditional machine learning methods presented thus far don't leverage this sort of spatial and temporal data to make better-informed decisions. To this end, Markov Random Fields are particularly helpful.

The data used for MRF was described in the “Data manipulation” section. With each tweet in this data set is information on the city from which the tweet’s user came from[1], and to simplify our model we mapped each city to its state, just to make the locations less granular.

Now, the goal for us here is to try to use location/time data to extract the kind of weather condition from a tweet. The general idea behind Markov Random Fields is to model dependencies between variables and gain insight from these dependencies when making predictions. To use this concept, we assume that close-together regions experience similar weather, and we focus on just a small window of time (half a day) to avoid vast variations in weather within a region.

The graphical model is as follows. We have a graph with nodes corresponding to states within the United States. There is an undirected edge between nodes if the corresponding states share a border. For each edge $e = (u, v)$ we assign a 15×15 table T_e, where the rows and columns are the 15 different weather kind categories – the rows correspond to state u and the columns correspond to state v (without loss of generality), and the value in table entry $T_e[i, j]$ is relatively large if there’s a high likelihood of state u experiencing weather i when state v experiences weather j, but relatively small if the likelihood of this is low. Given our assumption that neighboring states have similar weather, this means that values along the main diagonal will be larger than values off the main diagonal.

The goal is to determine the kind of weather for each node (i.e. each window of time for each region), and then label each tweet from that location/time with the kind of weather for the corresponding node. So the question is, how does one label a node? Consider a node u with, say, four neighbors n_1, n_2, n_3, n_4 which have already have labels l_1, l_2, l_3, l_4 fixed. We also already have a training set of tweets from the location/time corresponding to node u, and we collect the words from all these tweets into a bag of words; for each word, we have a length-15 vector $\bar{w}$ of probabilties $P(y|w)$, i.e. the probability that the kind of weather is y (for all 15 kinds of weather) given that a tweet has word w.

Then, to label a node, we do the following. For each neighbor n_i there is an edge matrix $T_{(n_i,u)}$ as defined earlier, and since the label of n_i is fixed, we have a row vector (could be a column vector, but I’ll call it a row without loss of generality) corresponding to the likelihood of u having a certain label given that n_i has label l_i. Let’s normalize these vectors so that the entries are probabilities.

Additionally, for each tweet we’re classifying from u’s location/time, we have a vector of probabilities that the tweet belongs to each kind of weather (from Naive Bayes). Multipliying these tweet vectors and neighbor vectors pointwise, and then normalizing, we produce a final vector v of probabilities that node u belongs to each particular kind of weather – these probabilties therefore take into account the training data *and* the location/time, as desired. Using this vector of probabilities, we draw at random a kind of weather to label u with (the random drawing is done in a way so that the probability of drawing the i^{th} kind is v_i).

Repeating this process for each node gives us labels for each node. This begs the question – how do we come up with an initial labelling of nodes? Certainly, some initial labelling is necessary in order to assume that the neighbors are pre-labelled. To this end, we require the method Gibbs Sampling. We assign each node an initial label at random (or not-so-random, e.g. an educated guess given prior results from Naive Bayes), and then perform our labelling as described. After several (perhaps hundreds) iterations of labelling each node, we’d converge to a label likely to be correct. At this point, we get a label l for each location/time node u by taking the mode of all labels produced for u throughout the Gibbs Sampling, and we assign label l to each tweet from the location/time corresponding to u.

[1]It could be the case that a user from city A is actually making a tweet from city B and commenting on city B’s weather, but the hope is that this isn’t too frequent.

One slight shortcoming here is that we assign each tweet one kind of weather, when a location can experience multiple kinds of weather at one (e.g. cloudiness and rain). A solution here is to look at the final probability vector for a node and label the node with the label of max probability plus all labels with probablilities greater than or equal to some threshold (e.g. 0.7).

3.3.1 Markov Random Fields (MRF)

4 Experimental Evaluation

4.1 Results

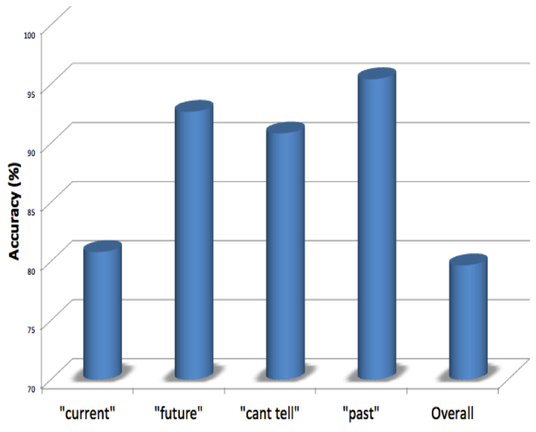

Figure 4: "Sentiment" Accuracy

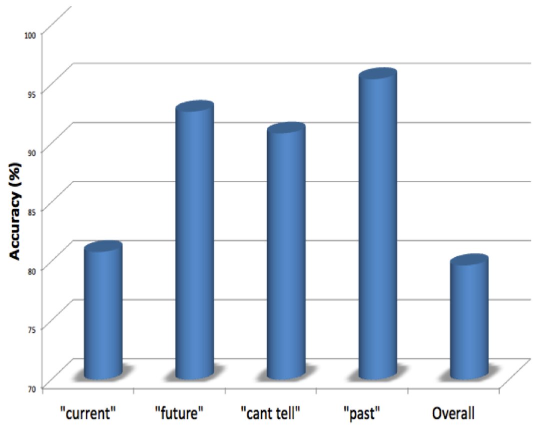

Figure 5: "When" Accuracy

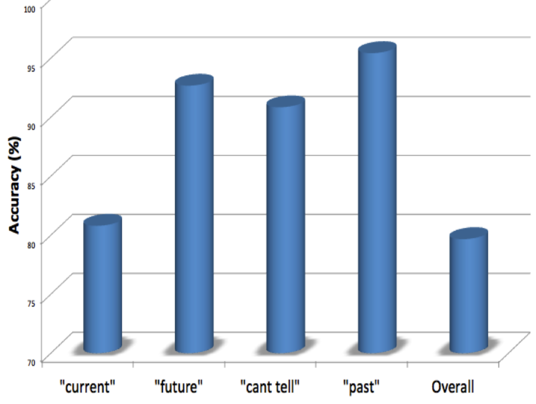

Figure 6: Average "Kind" Accuracy

4.2 Discussion

5 Related Work

Out of the 4 algorithms we tried (SVM, Naive Bayes, Decision Tree and Markov Random Fields), 2 of them were Generative algorithms and 2 were Discriminative. Our Markov Random Field additionally used the conditional probabilities based on time and location of the tweets to give a similarity measure to the prediction. The training set we originally had from the

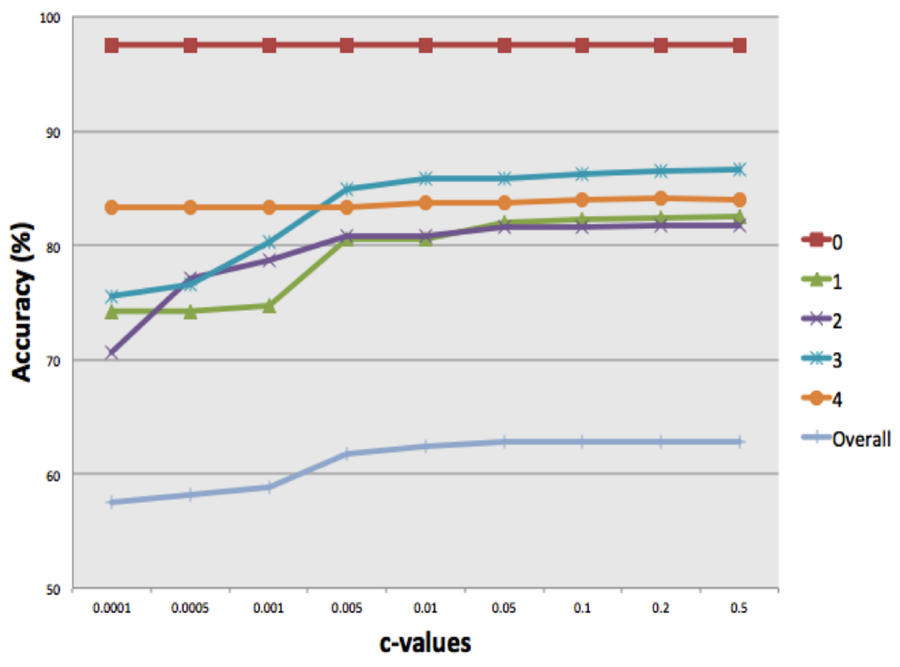

Figure 7: Accuracy vs. C-value

n-Gram Blend			"Sentiment" Accuracy	"When" Accuracy	Average "Kind" Accuracy
Unigrams	Bigrams	Trigrams			
1	0	0	57.710	76.935	87.376
0	1	0	34.322	43.706	46.524
0	0	1	15.074	17.710	18.307
1	1	0	60.936	78.568	88.479
1	0	1	61.582	78.256	88.179
0	1	1	32.293	63.644	72.396
1	1	1	62.536	77.743	87.802

Figure 8: Accuracy Comparison of n-gram Blends

Kaggle competition did not have timestamps on the tweets. This proved to be a problem for us in implementing the MRF learner, as location similarity is not enough to give an accurate classification. For instance, two particular tweets could be from the exact same location, but 6 months apart in time, therefore having different weather.

The research we did in bringing this project to completion allowed us to read about multiple algorithms, or modifications to existing algorithms that would give us a very robust classifier. In particular, we read about various Probabilistic Graphical Models that made the use of a "grid-like" arrangement of the tweets, with edges between tweets representing a dependency. The Markov Random Field algorithm we implemented was one such Probabilistic Graph Model.

In these models, the basic idea was that the tweets had certain independent variables, and certain dependent variables. In our situation, the independent variables were the class-conditional word probabilities (inferred from the training set word counts) and the independent variables were the location and time-based weights. Another such algorithm was the Conditional Random Field, which made the use of a similar kind of pattern recognition for structured

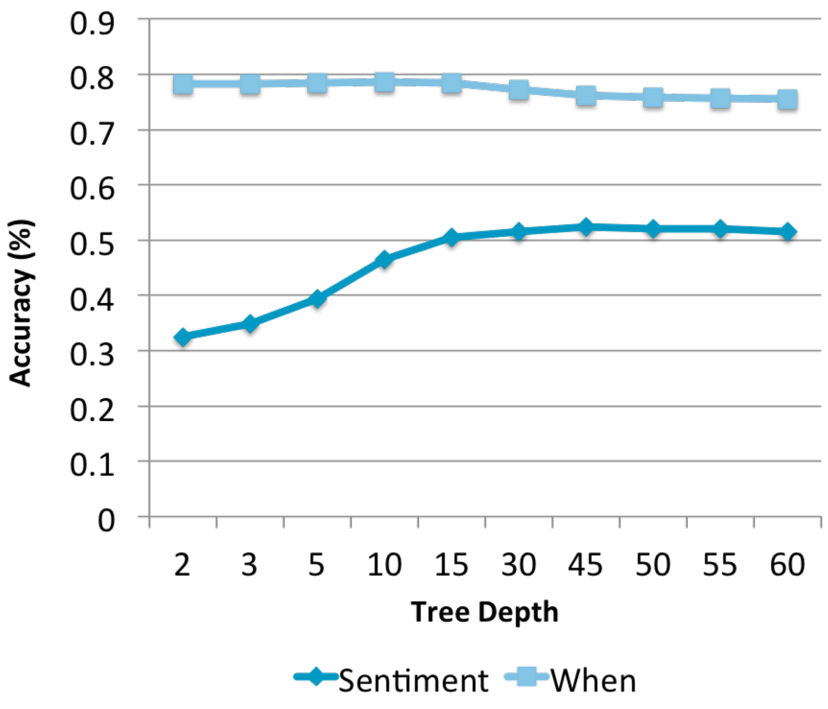

Figure 9: Accuracy vs. Tree Depth

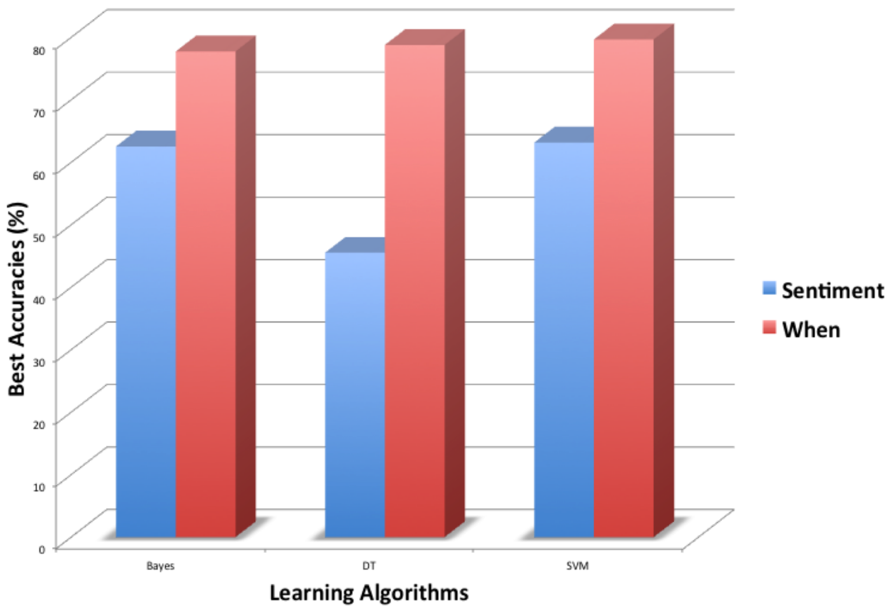

Figure 10: Comparison of Learning Algorithms

predictions based on “similar” tweets.

The problem with most of these algorithms ended up being that they made assumptions about a sequence of observations being related. However, in our case, we had tweets that might be within the same timeframe but not related due to location differences, or vice versa. We also began with a limited dataset that did not have this vital information; and even with the Cheng-

Caverlee-Lee dataset, we did not have latitudes and longitudes to introduce a proper distance-based similarity, but instead we had city and state names. Keeping in mind the constraints on our datasets, the Markov Random Field was the most appropriate to implement, using the conditional-dependencies as a Probabilistic Graphical Model.

6 Future Work

There are a few ways in which we look to expand upon the aforementioned algorithms:

- Loosening the requirement of linear separability for SVMs by incorporating kernels
- Giving different weights to unigrams, bigrams, and trigrams in Naive Bayes
- Adjusting weights (importance) of certain words in Naive Bayes, e.g. giving additional weight to emoticons and hashtags
- Full implementation of MRF, performed for several different time windows and with different sets of edge matrices T
- Modeling dependencies between sentiment and weather kind (e.g. if sentiment is negative, then it's more likely to be snowy than sunny)

7 Conclusion

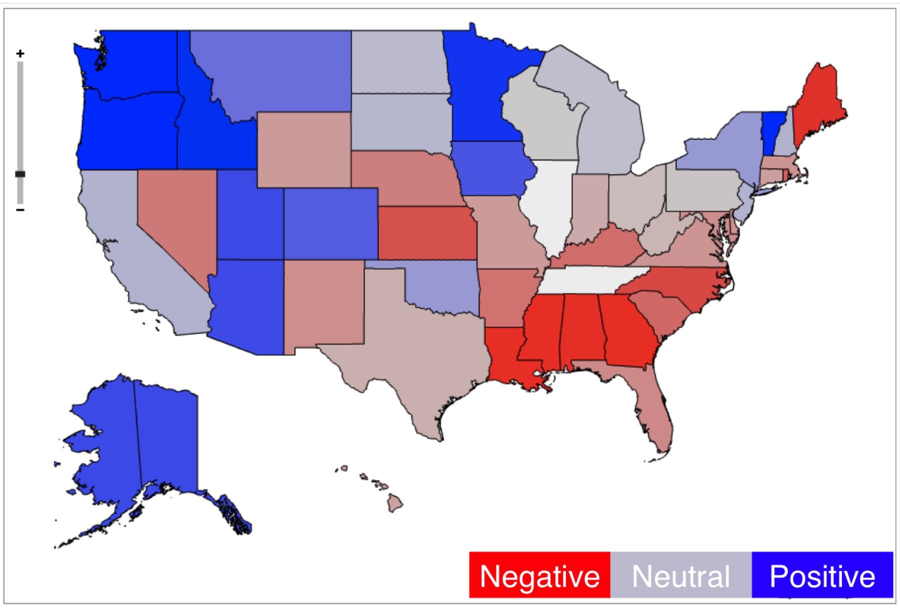

Figure 11: Average "Sentiments" of the United States

8 Appendix

9 Conclusion

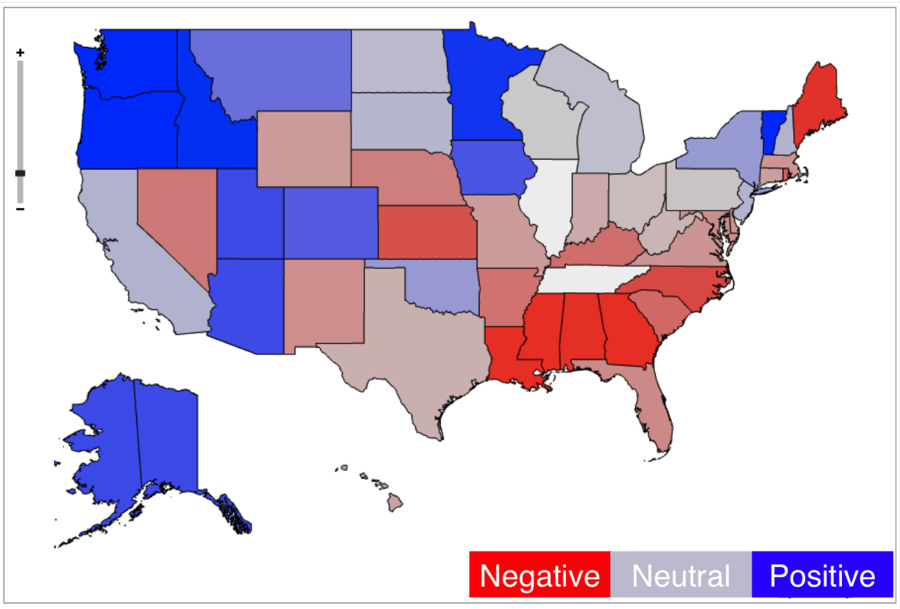

Figure 12: Average "Sentiments" of the United States