Extracting Emotions from News Headlines

Raghavan, Bharad
bharadr@stanford.edu

Samar, Anshul
asamar@stanford.edu

December 13, 2013

Abstract

Determining emotion in a piece of text is a difficult subset of sentiment analysis, a field which has largely focused on binary classification of text as positive or negative. Classifying sentences in one of six major emotion categories (anger, disgust, fear, joy, sadness, and surprise) can make significant contributions to political polling, marketing, and stock market predictions. In this study, we focus on the first of these six emotions, anger, and develop a model to predict whether or not a reader will feel anger after reading a headline. Our model involves NLP tools - stemming, named entity recognition, and part of speech tagging - and a bag of words model run on a SVM with a linear kernel and a Naive Bayes Classifier with Laplace smoothing. We developed experimental models to test on smaller subsets of data including: PCA, to reduce dimensionality of data, a synonym algorithm, to expand each headline allowing our model to classify with respect to more words, and a co-occurrence model, to score each headline based on co-occurrences of words with the word 'anger' in the New York Times archives. We also made an emotion headline generation model with Markov chains to see if we can automatically create angry headlines. With these models, we hope to gain greater insight into methods that one can use to detect emotion within text.

1 Overview

In this paper, we discuss our original model, provide error diagnostics, and describe our final model built based on the lessons we learned. We also discuss our experimental additions and possible next steps.

2 Data Set and Pre-Processing

Initially, we used SemEval's data set [1] of 1250 news headlines as our testing set (from now on, referred to as 'SemEval'). SemEval contains headlines scored from 0 to 100 for six different emotions. We provided a threshold on which to classify each headline (headlines that had been scored by SemEval as greater than 20 were interpreted by our system '1' - i.e. containing a non-negligible amount of anger).

However, we had no training data. We created a lexicon we believed instigated anger, based loosely on a WordNet anger lexicon [2]. Our lexicon included words like 'anger', 'kill', 'accused', 'rape', etc. We wrote a script to extract headlines from The New York Times Article Search V2 API that included these words. We similarly extracted clearly non-Angry headlines with another lexicon of words associated with positive emotions. We hoped that in doing so, our model would pick up on words associated with anger and non-anger.

We developed a parser based on NLP tools to sanitize and normalize headlines. Among other features, we removed headlines that weren't properly encoded in UTF-8 and stripped them of hyphens/punctuation marks. We used the Stanford Named Entity Tagger to recognize locations and ran them across a lexicon of US states and added a feature as to whether or not the headline concerned domestic or international affairs (we theorized that headlines that included non US states like foreign countries were related to negative news). We used the Stanford Part of Speech (POS) tagger to remove prepositions. We detected numbers using POS and added features for that as well (we theorized that numbers are correlated with emotion). We removed articles and also resolved possessions - the Stanford tagger split possessions into the original word and the apostrophe s's were removed by our parser.

To normalize the headlines, we un-capitalized words and ran all of our headlines through NLTK's Snowball Stemmer (this can be tested online - 'thankfully, 'thankful', 'thanking' for example are turned into 'thank'). We then parsed each headline and returned a text file of unigrams and bigrams. We did this as

[1] http://www.cse.unt.edu/~rada/affectivetext/
[2] http://www.cse.unt.edu/~rada/affectivetext/

'happy' and 'not happy' reflect two different sentiments and the latter sentiment can be more easily captured through bigrams.

Note that the parser does not work with full accuracy as many of the tools it relies on are also based on machine learning methods.

3 Initial Model

We wrote scripts to extract XML data from SemEval for our test set as well as create a training set in Matlab from our accumulated NYTimes headlines. We created a lexicon and ran a BOW (bag of words) model. Because this is a text classification problem, we used a support vector with a linear kernel (referred to as SVM) and a Naive Bayes classifier with Laplace smoothing (referred to in this paper as 'Naive Bayes') based on code from CS229 exercises. The SVM code came from liblinear.

4 Initial Results

Here were our results for the SVM (testing set is SemEval). n/m refers to n angry headlines and m non-angry headlines used for training.

Angry/NonAngry	Precision	Recall	Accuracy
500/1000	28.12	37.97	7.2
1000/1000	25.12	66.24	12.56
1500/1000	24.5	77.64	14.72
2000/1000	23.52	78.9	14.96

Here were our results with Naive Bayes (testing set is SemEval).

Angry/NonAngry	Precision	Recall	Accuracy
500/1000	33.81	59.49	70.24
1000/1000	25.3	80.17	51.36
1500/1000	22.18	90.3	38.08
2000/1000	21.46	94.51	33.36

5 Error Diagnostic and Analysis

Analysis for SVM We experimented with data distributions for our training set. The SVM worked best with a 2000/1000 ratio. As ratios of angry to non-angry headlines increased, accuracy and recall went up, while precision and recall went down. We performed further tests (not shown above) in which we increased the size of our dataset while keeping to a 2:1 distribution - but the precision remained relatively consistent (hovering between 22 - 23%), while recall and accuracy decreased. When we performed an error diagnostic on 500/250 headlines, we found that our SVM had 87% recall on angry headlines but classified 67.3% of non-angry headlines as angry.

Analysis for Naive Bayes Naive Bayes worked best with 500/1000 distribution. As the ratio of angry to non-angry headlines increased, recall increased sharply at the expense of precision and accuracy. Our Naive Bayes on 500/1000 distribution classified 59% of angry headlines as angry, but also classified 27.2% of non-angry headlines as angry. The Naive Bayes has higher accuracy and recall than SVM, however levels of precision are relatively similar.

Conclusions:

1. **Distribution**: Testing results depend significantly on distribution (an excess of angry headlines leads to more words being indicative of anger which leads to higher recall). Our results are low simply because the training and test data are drawn from different distributions! One is created automatically from our API and the other from SemEval. Our dataset does not capture the true distribution of anger in news headlines.

2. **Bag of Words**: We theorize that because BOW does not take into account ordering/semantic meaning, we classify a lot of non-angry headlines as angry because they contain the same words as angry headlines. For example, one might be angry at the headline 'Taliban kills American troops' but not angry at 'American troops kill Taliban.'

3. **Data**: Even though our initial results might make it seem that having smaller data sizes is better, we know that having more data is preferable as its leads to a larger lexicon and thus more words in the testing data are being captured. Furthermore we found that in the SVM on 500/250 only 12.59% of words in the test lexicon appeared in the training lexicon. Meanwhile, in Naive Bayes on 500/1000 only 17.64% of words in the test lexicon appeared in the training lexicon. After discussions with a TA, we learned that most text classifications are done with massive corpuses that are greater than the number of words in the English lexicon. This clearly is a problem for us and is something to consider in the future.

4. **Naive Bayes vs. SVM**: Our Naive Bayes does better than SVM on accuracy and recall and has comparable results on precision. The reason for this is because the angry headlines in our dataset contain key words from our lexicon. Because Naive Bayes assumes independence on all

features it found a strong correlation between appearances of these angry words and having anger in the headline. In the SVM however, these features are not considered independently.

5. **Recall**: Naive Bayes classifies less non-angry headlines as angry. We gave Naive Bayes a dataset of 500/1000 and it classified 27.2% of non-angry headlines as angry. We gave SVM a dataset of 500/250 and it classified 67.3% percent of non-angry headlines as angry. We performed this error diagnostic on the distributions in which each algorithm did best. We think these results are due to our flawed distributions - the Naive Bayes was given less angry headlines to begin with and thus did not have a high tendency to label non-angry headlines as angry. The SVM had a flipped distribution and thus labeled more non-angry headlines as angry.

6 New Dataset

We decided to create a new dataset from which we would do cross validation. With this, we could ensure that both our testing and training data came from the same distribution.

We self-labeled New York Times headlines from December 1st 2013 to December 10th 2013 for anger/nonAnger. We also included headlines from our API calls, but made sure to remove words from our lexicon which we had originally used to query. We did this because those words were obvious indicators of an angry headline. For example, we turned our automatically generated headline 'berlusconi accused of bribing witnesses...' into 'berlusconi of bribing witnesses...'. Lastly, we included SemEval's dataset within this training set. Our data pool thus had 4527 angry headlines and 4668 non-angry headlines.

We conducted a random sampling of 100 headlines from our automatic NYTimes collection to see how many were actually anger related (our parser removes some headlines, but this is still a good estimate of the error). From non-angry headlines, 7% were indicative of anger. From our angry headlines, 17% were indicative of non-anger. While there is some degree of corruption in our automatically generated training set, the majority are still fairly well labeled. Note that these corrupted headlines are supplemented with human annotated headlines from SemEval and headlines from the month of December.

7 New Results

With our distribution problem solved, we fed into our algorithms a 1:1 ratio of angry to non-angry headlines. We randomly permuted our data set and ran cross validation - training on 90 percent of our set and testing on 10 percent. We then averaged the results of 10 trials. Here were our results with SVM doing cross validation.

Angry/NonAngry	Precision	Recall	Accuracy
500/500	70.85	71.91	71.2
1000/1000	72.66	73.82	73
2000/2000	70.49	74.06	71.55
4000/4000	71.77	74.58	72.61

Here were our results with Naive Bayes doing cross validation.

Angry/NonAngry	Precision	Recall	Accuracy
500/500	72.48	73.18	72.5
1000/1000	74.97	76.07	75.25
2000/2000	71.17	78.48	73.32
4000/4000	71.94	75.02	72.88

8 Analysis

After choosing to run cross-validated data with the same distribution, our numbers for the both classifiers increased dramatically across the board. Accuracy, precision, and recall were above 70% in both cases, and the performance of the both algorithms remained consistent even as the size of the dataset was increased. The recall of Nave Bayes was slightly higher than that of the SVM (by approximately 1-4%), but in all other aspects both approaches yielded comparable results.

Analysis: We trained and tested on the same data pool - this by itself improved the performance of our algorithms as both training vectors and testing vectors are drawn from the same distribution. Also note that when compared to our old data set, the recall for Nave Bayes and SVM was not as high. This is partly due to our new dataset having an equal amount of angry and non-angry headlines (as opposed to having an excess of one over the other, leading to over-classification on one of the two outcomes). We think this was also because the automatically extracted portion of our data set was stripped of angry 'keywords we used to query the NYTimes API. This prevented the Nave Bayes classifier from identifying the obvious correlations between the appearance of angry keywords in headlines and the headline being classified as angry, and thus made for a less skewed training set (and leads to a drop in recall).[3]

[3]Note that with API calls and parsing there may be some inaccuracies (jumbled words, missed headlines, etc). Some headlines still contained lexicon words.

9 Experimental Models

We decided to experiment with models to fix some of our recurring problems: lack of a large lexicon, limited semantic analysis, and large dimensionality.

Here were our results on SVM with 500/500 from our data set with PCA and cross validation.

Model	Precision	Recall	Accuracy
No PCA	70.85	71.91	71.2
PCA	71.03	75.56	72.3

PCA [4] We had more features than data points and thus decided to run PCA to see if it could determine common subspaces which held training points. We performed PCA on 500/500 training set, and ran the SVM on the reduced training matrix. The number of features we had in our feature vector had an avg. of 6306 features and after PCA this number was reduced to an average of 775 features. Due to the sparseness of our training matrix and incompleteness of our lexicon, finding common subspaces for all the data points was likely difficult, but the slight increase in recall and accuracy might be attributed to the elimination of noisy data when making the reduced matrix.

Here were our results on SVM with 200/200 from SemEval with experimental models (described below) + cross validation.

Model	Precision	Recall	Accuracy
Normal	73.73	75.46	74
Synonym	79.55	85.53	82
Co-Occurrence	74.72	75.87	75
Synonym/Co-O	76.2	74.58	75

Here were our results on Naive Bayes with 200/200 from SemEval with experimental models (described below) + cross validation.

Model	Precision	Recall	Accuracy
Normal	68.71	78.44	71
Synonym	82.38	92.04	86.25
Co-Occurrence	79.46	69.15	74.75
Synonym/CoO	92.6	70.42	82.5

Synonym Model With limited data, we barely cover all 'similar' words in the English lexicon. Hence, we decided to expand headlines with synonyms to be able to 'capture' more words across fewer headlines. Unfortunately we were only able to expand 150 angry headlines because of API call limits, but even with this we showed significant improvements. (Note that due to API call limits for synonym expansion, we only applied this model to a small data set - specifically the first 200 angry and non-angry headlines from SemEval).

We made very significant improvements with synonyms, with a 15.25% increase in accuracy for Naive Bayes. We had 8% increase in accuracy for SVM.

Co-Occurrence Model One of the techniques we saw in research papers looked at the correlation of words with words representing an emotion. We wrote a script to remove uninteresting words (prepositions, 'the', 'be', etc) from headlines and then queried the NYTimes API for all the headlines between Jan 1st, 2003 and December 10th, 2013 that contained a word in the headline and the word 'anger.' We scored each word with the count of headlines from the API call and summed the counts together, dividing it by the number of words in the headline (for normalization). We added this to our training matrix and ran SVM and NB on it. Due to limited time, we only applied this model 200 angry and non-angry headlines from SemEval.

We made moderately significant improvements with the co-occurrence model, with a 3.75% increase in accuracy in Naive Bayes and 1% accuracy in SVM.

10 Intelligent Headline Generation with Markov Chains

We attempted to generate angry headlines by supplying a "seed word" to our program based on a Markov Chain model. We took in a corpus of angry and non-angry headlines and kept track of all words that could succeed or precede any given word. Our output headlines were 6 words long and the seed word appeared anywhere in the headline. The words preceding and succeeding the seed word were randomly picked from a list of generated successors/predecessors.

One would think this could be modeled as a chain factor graph where we assign words to maximize the potentials - i.e. the transition probability between 2 words). But since the domain of the nodes (possible words one could place in one of the 6 places on the headline) is completely dependent on the value of the node beside it (domain would be the list of possible successors/predecessors of a given word), we cannot use a constraint satisfaction problem (CSP) solver to maximize the potential. Instead we randomly generated hundreds of headlines with the given seed word and rank them in descending order by coherency (which is the product of all the transition probabilities between the words in the headlines). These headlines are then classified as angry or not angry and returned for us to judge. In this case, we used our Naive Bayes classifier (trained on 1000/1000 from our data set) to determine whether or not the headlines were angry.

[4] Code based on tutorial from: http://matlabdatamining.blogspot.com/2010/02/principal-components-analysis.html

Here are some examples of generated headlines for Anger (Seed Words: Rape or Kill): Prudery Blasts Kill Dozens Beginnings Asia Have, Temple Terry Blasts Kill Dozens Deception Nelson, It Simple in Rape of Genocide Scenes, Rape Problem Solving Fruit Fleiss Theft of.

While the classifier is able to pinpoint headlines containing words commonly found among angry headlines, its inability to look at word ordering or semantics makes it classify nonsensical headlines as angry. Another flaw is how the Markov Chain program measures coherency between every pair of words instead of the overall sentence.

11 Conclusions and Next Steps

In conclusion, we found that our training set, distribution, and experimental models significantly contributed to our results and the success of this project. Whether our pre-processing step and development of a parser to stem and identify parts of headlines, collection of a data set through automatic NYTimes API scripts, usage of SMV and Naive Bayes, and experimental models with synonyms, co-occurrence, and PCA, we learned that many parts of the machine learning pipeline contribute to the final result - not just the algorithm itself.

Due to time and resources, we were not able to implement some of our ideas. Here are some of them:

1. Run the experimental models on more data.
2. Get more features from NYTimes including lead paragraphs
3. Sort the words by 'importance' (for example in 'Girl killed in Egypt' the most important word is killed). We experimented with Stanford's online root word finder, however for the headline, 'Rise in Deadly Attacks on Shiites in Iraq Stirs Anger at Government' it gave us 'rise' as the root word. We would need to find a better way of finding critical words in headlines.

12 Research

We read quite a few papers, here were some of the ones we read that were useful. Yang et a. [4] discussed CRF (conditional random field) to assist classification. It locally searches for emotion in the vicinity of the sentence it is classifying. Wang et al. [3] automatically collected data through twitter hashtags like #annoying or #nervous and applied POS and n-grams. This made us think about increasing our data set and using POS. Strapparava and Mihalcea's paper [2] uses SemEval and applies Latent Semantic Analysis which determines similarities (and can also use "synonyms" of words as well as entire emotion synsets). Kozareva and Navarro [1] also worked with SemEval and used mutual information across webpages on WWW to determine emotion - in general terms, they determined whether particular words in headlines were found in pages with particular words in a known emotion lexicon. This was similar to our co-occurrence model.

13 Acknowledgements, Notes to Staff

Please note that both of us are new to NLP and have not taken a language processing class before. Bharad Raghavan is in both CS221 and CS229 and is using this for his final project for both classes. Anshul Samar is in CS229.

Acknowledgements include: CS229 Class, CS221 notes, TAs from CS229/CS221, Stanford parsing tools, classmates for debugging, matlabdatamining.blogspot.com their tutorial on PCA, Callback Js Javascript class for making API calls, Thesaurus service was provided by words.bighugelabs.com, and many online forums for resources.

Lastly, note that originally we were going to classify for all six emotions (and we did that for our milestone). However, we decided that it would be best to focus on 'anger' as we spent time creating our own data set (and doing so manually for all six would have been too much given the time we had).

We have made all code public at github.com/anshulsamar/pathos.

References

[1] Zornitsa Kozareva and Borja Navarro. UA-ZBSA: a headline emotion classification through web information. *Proceedings of the 4th ...*, (June):334–337, 2007.

[2] Carlo Strapparava and Rada Mihalcea. Learning to identify emotions in text. *Proceedings of the 2008 ACM symposium on Applied computing - SAC '08*, page 1556, 2008.

[3] Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P. Sheth. Harnessing Twitter "Big Data" for Automatic Emotion Identification. *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 587–592, September 2012.

[4] Changhua Yang, Kevin Hsin-Yih Lin, and Hsin-Hsi Chen. Emotion Classification Using Web Blog Corpora. *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*, pages 275–278, November 2007.