COMP 4106

FINAL PROJECT REPORT

Geetika Sharma
(100993465)

TWITTER SENTIMENT ANALYSIS

1. PROBLEM STATEMENT

Twitter serves as a mean for individuals to express their feelings and thoughts on different subjects. They have been analyzed by several marketers to conduct market analysis. I am conducting Twitter sentiment analysis using some of the machine learning techniques. I attempt to classify the tweet as either positive or negative, where both are present, the more dominant one shall be picked. The aim of the project is to compare some of the classification methods and neural networks in terms of accuracy.

2. STATE OF THE ART

The State of the Art, which is also the motivation of this Project is the Stanford NLP Sentiment Analysis tool, which uses a Recursive Neural Network that build on top of grammatical structures. This model is unlike all the others in use today, and that is simply because we analyze words in isolation for prediction, but this model computes the sentiment based on how words compose the meaning of longer phrases.

3. DATA DESCRIPTION

I am using Data Set from Kaggle, which is separated into Test (300000) and Training (100000) Sets in the .csv form. The dataset is a mixture of words, emoticons, symbols, URLs and references to people and emoticons contribute to predicting the sentiment, but URLs and references to people don't, therefore, URLs and references can be ignored. The words are also a mixture of misspelled words, extra punctuations, and words with many repeated letters. The tweets, therefore, have to be preprocessed to standardize the dataset.

The Pre-processing done is as follows:

- Convert tweet to lower-case
- Replace multiple dots(.) with a space
- Strip spaces and quotes from the end of tweets
- Replace multiple spaces with a single space
- Converting 2 or more letter repetitions to 2 letters
- URLs are matched ((www\.[\S]+)|(https://[\S]+)) and replaced with word URL
- Twitter handles are matched with @[\S]+ and replaced with word USER_MENTION
- Hashtags are matched with #(\S+) and removed.

– Emoticons are processed as:

Emoticon(s)	Type	Regex	Replacement
:), :), :-), (:, (:, (-:, :')	Smile	(:\s?\)|:-\)|\(\s?:|\(-:|:\'\))	EMO_POS
:D, : D, :-D, xD, x-D, XD, X-D	Laugh	(:\s?D|:-D|x-?D|X-?D)	EMO_POS
;-), ;), ;-D, ;D, (;, (-;	Wink	(:\s?\(|:-\(|\)\s?:|\)-:)	EMO_POS
<3, :*	Love	(<3|:*)	EMO_POS
:-(, : (, :(,):,)-:	Sad	(:\s?\(|:-\(|\)\s?:|\)-:)	EMO_NEG
:,(, :'(, :"(	Cry	(:,\(|:\'\(|:"\()	EMO_NEG

```
[Analysis Statistics]
Tweets => Total: 100000, Positive: 56462, Negative: 43538
User Mentions => Total: 88955, Avg: 0.8895, Max: 12
URLs => Total: 3599, Avg: 0.0360, Max: 4
Emojis => Total: 1184, Positive: 997, Negative: 187, Avg: 0.0118, Max: 5
Words => Total: 1192638, Unique: 50378, Avg: 11.9264, Max: 40, Min: 0
Bigrams => Total: 1093170, Unique: 385115, Avg: 10.9317
```

Figure 1. Statistics of processed training Set

A tweet is represented as a feature vector as a sparse vector representation or a dense vector representation based on the classification method being used. For Sparse Vector Representation I used two types of feature representations, presence and frequency; where for presence feature type the feature vector has 1 at indices of unigram and bigram and 0 elsewhere, and for frequency feature type the feature vector has a positive number ay indices of unigram and bigram and 0 elsewhere. Dense Vector Representation an index is assigned to each word depending on its rank, i.e. the most common word gets index 1 and so on. Each tweet is then represented by a vector of these indices which is a dense vector.

4. DETAILS OF SIMULATIONS USED

a. NAÏVE BAYES CLASSIFIER

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set.
I used the MultinomialNB from sklearn.naive_bayes package of scikit-learn for Naïve Bayes Classification, which is useful for classification with discrete features. I used Laplace smoothing of Naïve Bayes with α =1. I used sparse vector representation and ran simulations using both presence and frequency feature type, and eventually found the presence feature type to outperform frequency, which is probably because

Naïve Bayes works better on integers then frequency distribution. I initially ran the simulation using Unigrams alone, but when I added Bigrams the accuracy increased to a 79.68% as opposed to 78.16% for unigrams alone.

b. MAXIMUM ENTROPY

Maximum Entropy Classifier model is based on the Principle of Maximum Entropy. The main idea behind it is to choose the most uniform probabilistic model that maximizes the entropy, with given constraints. Unlike Naive Bayes, it does not assume that features are conditionally independent of each other. So, we can add features like bigrams without worrying about feature overlap. The nltk library provides several text analysis tools.

I used the MaxentClassifier to perform sentiment analysis on the given tweets. Unigrams, bigrams and a combination of both were given as input features to the classifier. I first ran the simulation with GIS Algorithm which gave an accuracy of 79.34% for Unigrams alone and 79.2% for Bigrams. I then changed the Algorithm to IIS and used feature combination of unigrams and bigrams to get an overall accuracy of 80.98%.

c. SUPPORT VECTOR MACHINES

SVM, also known as support vector machines, is a non-probabilistic binary linear classifier. For a training set of points $(x_i; y_i)$, where x is the feature vector and y is the class, we want to find the maximum-margin hyperplane that divides the points with $y_i = 1$ and $y_i = -1$.
I utilized the SVM classifier available in sklearn, with C = 0.1(the penalty parameter of the error term). I initially ran the simulation with Unigrams alone and then Bigrams alone, which gave me 79.54% and 79.83% accuracy respectively. I then ran a combination of the two using presence which increased the accuracy to 81.11%. The best result was obtained by using frequency feature type on combination of both unigram and bigram.

d. CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks or CNNs are a type of neural networks which involve layers called convolution layers which can interpret spacial data. A convolution layers has a number of filters or kernels which it learns to extract specific types of features from the data. The kernel is a 2D window which is slided over the input data performing the convolution operation. I used temporal convolution ,which is suitable for analyzing sequential data like tweets.
I used keras with TensorFlow backend to implement the Convolutional Neural Network model and trained the model using dense vector representation, using the top 90000 words from the training dataset. I used the Sequential keras model, where

the first layer needs to specify the input shape. So, the first layer is the embedding layer which is a matrix of shape (v+1) *d where v is vocabulary size (90000) and d is dimension of word vector(200), and initialized with random weights from N(0, 0.01). Each row of the Embedding matrix represents the 200-dimension word vector for a word in the vocabulary. For words that match the Stanford glove200d file, I seed the corresponding row of the embedding matrix from the GloVe vectors. I trained the model using binary cross entropy loss using Adams's optimizer(which is used because the model converges faster).
The simulation model consists of a 4-layer convolutional network. The model architecture is as follows:
embedding_layer (90000_ 200) → dropout (0.4)
→ conv_1 (600 filters) → relu → conv_2 (300 filters) → relu ! conv_3
(150 filters) → relu → conv_4 (75 filters) → relu → flatten → dense(600) → relu → dropout(0.5) → dense(1) → sigmoid

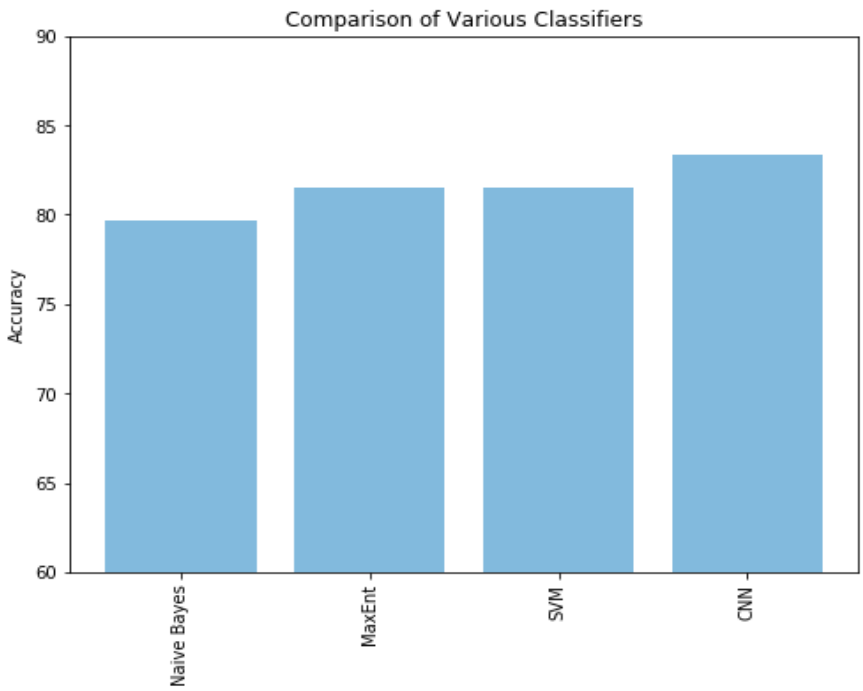
After simulations I found out that using Flatten as opposed to MaxPooling and increasing the number of units in the fully connected layer helped the model learn better. As the convolution layers increased so did the accuracy, with 82.40, 82.76, 82.95 and 83.34 for 1-layer, 2-layer, 3-layer and 4-layer network respectively.

5. SUMMARY

I used Naïve Bayes, Maximum Entropy, SVM and CNN to classify the polarity of the tweets. I used two types of features namely unigrams and bigrams for classification and observed that augmenting the feature vector with bigrams improved the accuracy. Once the feature has been extracted it was represented as either a sparse vector or a dense vector. It has been observed that presence in the sparse vector representation recorded a better performance than frequency.

Comparison of classifiers that used sparse vector representation

Technique	Presence		Frequency	
	Unigram	Unigram + Bigram	Unigram	Unigram + Bigram
Naïve Bayes	78.16	79.68	77.52	79.38
Maximum Entropy	79.96	81.52	79.7	81.5
SVM	79.94	81.11	79.83	81.55



"The non-linearity of the network, as well as the ability to easily integrate pre-trained word embeddings, often lead to superior classification accuracy." — A Primer on Neural Network Models for Natural Language Processing, 2015.

This is rightful remark because after working on this project I came to the same conclusion keeping in mind all the various statistics, the Convolutional Neural Network out-performs all the other classifiers.

6. DISCUSSION

One of the major problems still remain efficient memory management because I found my system crashing a lot of times until I came across *scipy.sparse.lil_matrix*, which is linked list based implementation for sparse matrices. I also tried to use Python generators instead of keeping the entire dataset in the memory wherever possible. The Convolutional Neural Networks are somewhat tricky because tuning the filter size and kernel size has considerable effect on the outcomes. I somewhat observed for most that there is a lot that can be done using this technique and would like to explore further. I also aim at incorporating more emotions in this model, initially beginning with the neutral emotion. I also am looking to implement an LSTM network to see how that changes the performance of the simulation. After considerable work, my final aim still remains using this to create a similar GUI tool as the Stanford NLP that identifies emotions based on the textual input it receives, only for a wider range of emotions.