Sarcasm detection in plain text using Tensorflow

Saturday, December 16, 2017 12:45 AM

1. Introduction
   Sarcasm is the use of words that mean the opposite of what you want to say especially to insult someone, to show irritation, or to be funny. The goal of this project is to identify sarcasm in plain text. Automated detection of sarcasm in text is a difficult task due to a lot of reasons, such as, the lack of context in which the statement was made, the lack of intonation, the lack of expressions and body language, the lack of knowledge about the person's character and personality, amongst others. All these things could be the reason because of which a particular statement was considered sarcastic. However, the project plans to exploit the property of a general sarcastic statement of possessing contrasting sentiments by using Natural Language Processing. The project aims at training a neural to detect if a given statement is a sarcastic or regular sentence.

2. Motivation
   The idea behind this project comes from a web application, http://www.thesarcasmdetector.com/, that takes the input of a sentence and detects whether it is sarcastic. According to the open source code for the aforementioned application, it is implemented using two machine learning models, namely Naive Bayes and SVM. This project aims to implement a similar classifier using a deep neural network. Also, this [0].

3. Source of Dataset
   The dataset is obtained from [1] which had tweets collected over a period from June-July 2014. The dataset consists of positive sentences(sarcastic) and negative sentences. It consists of about 25,000 clean sarcastic tweets and about 1,00,000 clean non-sarcastic tweets. The tweets were collected from San Francisco and New York to ensure the tweets are in English. The minimum length of a sentence in the dataset is 3. 1

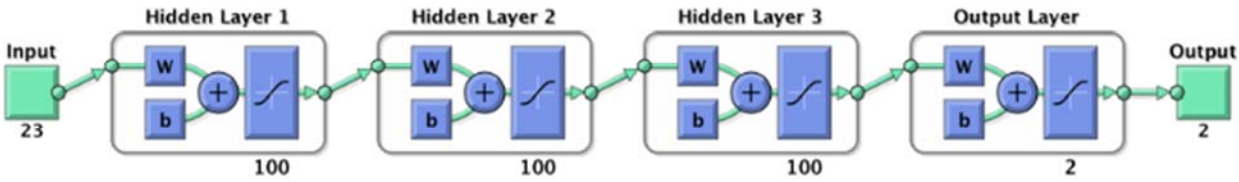
4. Feature Extraction:
   The motivation project uses three features of a sentence, namely n-grams, sentiments, and topics[2]. This was implemented using SentiNet and TextBlob, and gensim.
   In the current project, the polarity and subjectivity of a sentence is extracted using TextBlob.

   The following features are useful:
   a) Full sentence Polarity
   (b) Full sentence Subjectivity
   (c) Half sentence Polarity (1/2 and 2/2)
   (d) Half sentence Subjectivity (1/2 and 2/2)
   (e) Difference between polarities of two halves
   (f) One-Third sentence Polarity (1/3, 2/3 and 3/3)
   (g) One-Third sentence Subjectivity (1/3, 2/3 and 3/3)
   (h) Difference between maximum and minimum polarities of the thirds.
   (i) One-Fourth sentence Polarity (1/4, 2/4, 3/4 and 4/4)
   (j) One-Fourth sentence Subjectivity (1/4, 2/4, 3/4 and 4/4)
   (k) Difference between max and min polarities of the fourths.

   Polarity = positivity (-1 to 1)
   Subjectivity (0 to 1, 0 = objective, 1 = subjective).

   This results in 23 features being extracted and fed into the neural network as inputs.

5. Neural network architecture



A model with three hidden layers and one output layer was implemented with each layer consisting of 100 neurons. Although one output neuron is sufficient for a binary classifier, two were used in this case to allow for depicting of the output in terms of a one hot vector. The output layer has two neurons, one for each class, namely Sarcastic and Regular. A one hot encoded array makes it easier to extract and compare results.
The dataset was divided into 70% for the training set and 30% for the testing set.
Each layer multiplies the input provided to it with a set of weights and adds a bias to the value. The output of this step is sent through a ReLU activation function.

The cost is computed from the expected output and the predicted output using softmax_cross_entropy_with_logits. The softmax with logits function measures the probability error between the actual and the expected output and then squishes it such that sum of all elements is 1. Then cross entropy is used to calculate the error because it is proven to give a better measure of accuracy as opposed to the crude classification error. Then the mean of the output of the softmax function is calculated. This value gives the mean cost of the epoch. The goal is to train the neural network in such a way that the cost is minimized. It is achieved using the AdamOptimizer which uses stochastic gradient descent. The Adam optimizer is used for this because it provides faster convergence.

Instead of online processing, batch processing is used to train the network. The batch size for the input is chosen to be 50. The total number of epochs used for training is chosen to be 75.

Choosing different values for number of neurons, batch size, and number of epochs did not seem to make a considerable difference to the overall performance of the network.

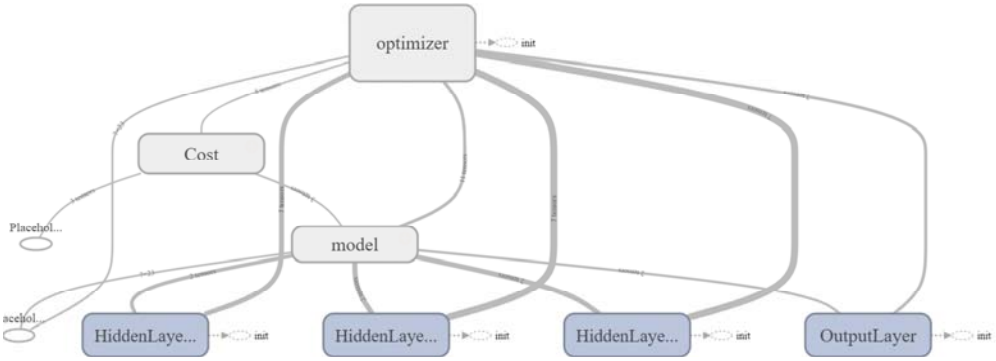
The metric used to guide cross validation is F-measure. This is due to the fact that our dataset is not balanced. Accuracy, as a metric, would fail, as a network which always classifies positively would return an accuracy of 83%. F-Measure is calculated using the values of precision and recall.

F-measure = 2 * ((precision * recall) / (precision + recall))

precision=true positives / (true positives + false positives

recall=true positives / (true positives + false negatives)

The tensorboard[3] visualization for the network is:



6. Experimental results

The existing website has an F1 score=.56 for n-gram feature extraction and .41 for sentiments extraction and .36 for topics extraction[2]. The proposed system is expected to have an F1 score of about 0.7, using only one type of feature. F-measure was computed using the scikit-learn package for python.

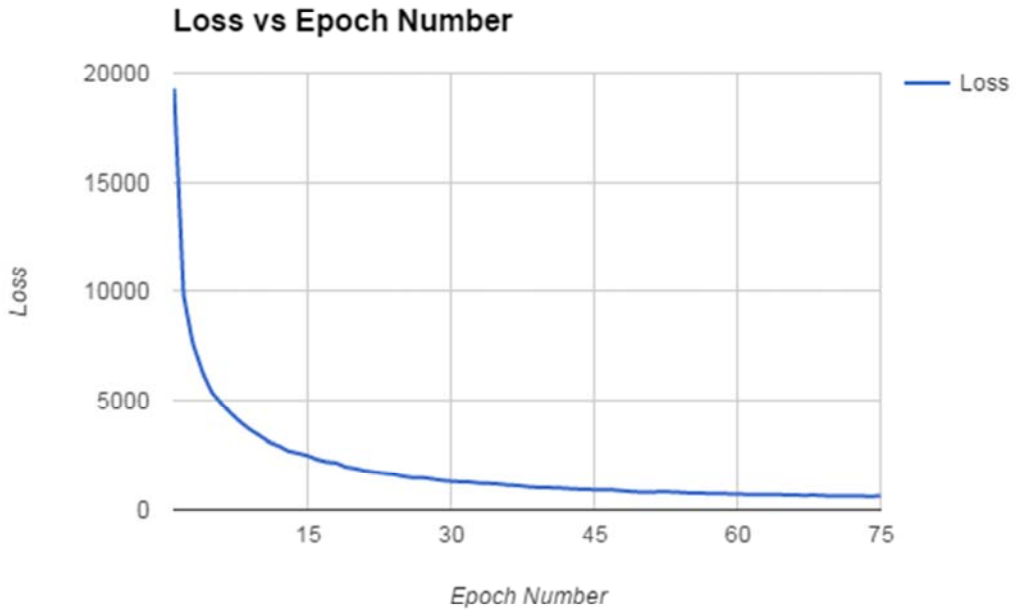
Sentence	Prediction
Going to the gym surely makes you fit, in a same way standing in a garage make you a car!	Regular
Nice perfume. Must you marinate in it?	Sarcastic
I am having way too much fun at this party.	Sarcastic

The confusion matrix of this binary classification:

Total=16419	Predicted: Sarcastic	Predicted: Regular
Actual: Sarcastic	2386	5199
Actual: Regular	1603	7231

F1 Score: 0.680116629044
Accuracy: 0.585724

The tensorboard[3] visualization for the learning process is:



7. Conclusion
   From the observations, it is evident that the project was able to obtain similar performances with the same dataset by implementing a deep neural network in TensorFlow with a lesser number of features. Although, if not compared to the motivation project, the accuracy can be considered to be low. This can be attributed to the limited size of the dataset

8. Requirements and procedure to run
   Python 3.5.x
   TextBlob 0.12.0
   Tensorflow 1.0.1
   Scikit-learn 0.18.1
   Scipy 0.18.1
   Numpy 1.12.1
   Nltk 3.2.2

   Procedure:

A. Run create_feature_sets.py; converts the sentences into features that are used by the neural network.
B. Run train_and_test.py; performs training over desired number of epochs and stores the weight and bias values for future use.
C. Run Use_NN.py after inputting a desired statement in the last line of the program source code; classifies the given statement into sarcastic/regular by using the stored weights.

9. Future work
   A much larger data set such as the Self-Annotated Reddit Corpus,[5] can be used to train the neural net. It consists of around 1.3 million sarcastic and 55 million non sarcastic statements. A similar feature extraction method can be used with the dataset.
   Use_NN.py can be modified to accept input sentences from the terminal window instead of having to modify the source code.

10. References

[0] BBC News, "http://www.bbc.com/news/technology-27711109", June 2014
[1] M. Cliche. The Sarcasm detector, ”https://www.github.com/MathieuCliche/Sarcasm_ detector", August 2014.
[2] M. Cliche. Why a sarcasm detector, ”http://www.thesarcasmdetector.com/about/r", August 2014.
[3]Visualizing learning,”https://www.tensorflow.org/get_started/summaries_ and_tensorboard".
[5] Self Annotated Reddit Corpus, "http://nlp.cs.princeton.edu/SARC/"