Sarcasm Detection in Plain Text Using Deep Learning with TensorFlow

Jayasankar , Savitha skj9180@rit.edu Khatwani , Sanjay sxk6714@rit.edu
Parekh , Saurabh sbp4709@rit.edu

21 August 2017

1 Abstract

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 machine learning model to detect if a given statement is a sarcastic or regular sentence.

2 Introduction

We found this unusual and fascinating project on the web called www.thesarcasmdetector.com[5]. It is a website that takes a sentence and tells you how sarcastic that sentence is. We found it so interesting that we started reading more and more about the topic and delving into the code of the project which is open source. The domain of the project completely aligned with what we had in mind for the project. Hence, we decided to take the same topic and implement it further.

The original project is implemented by two models; Naive Bayes and SVM, so we already had access to the results of those two models. In order to take it a step further we decided to implement a deep learning neural network, mainly because of their popularity and performance. We used Google's Deep Learning library TensorFlow[6] to build a deep neural network. This way, we have access to performance of three models for the same project.

3 Proposed Method And Implementation

1. Source of Dataset:

   The dataset is obtained from www.thesarcasmdetector.com which had tweets collected over a period from June-July 2014. The dataset consists of positive sentences(sarcastic)[3] and negative sentences[2]. 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.

2. Feature Extraction:

   Feature extraction is a crucial and time-consuming step when it comes to NLP projects. As the input is in text form, which cannot be directly given to the neural network as an input hence we need to extract features from the sentence. The base project has extracted multiple features from a sentence like n-grams, polarity using SentiNet and TextBlob, and the topic of the sentence using gensim[4].

   (a) N-Grams: N- grams are a contiguous sequence of n-words or n-items of a large sentence. Basically, unigrams and bigrams were used. Consider the sentence It is an awesome day. so if n=2 then bigrams would be
      i. It is
      ii. Is an
      iii. An awesome
      iv. Awesome day

   (b) Sentiment Analysis: The base project extracted sentiment as follows:
      i. Get polarity and subjectivity of the entire sentence.
      ii. Divide the sentence into two equal parts and then calculate polarities and subjectivity for those.
      iii. Divide the sentence into three equal parts and then calculate polarities and subjectivity for those.

      Two libraries were used to extract sentiments namely, TextBlob and sentiNet.

   (c) Topic extraction:- In topic extraction most relevant information of the sentence is extracted. It is implemented using python library gensim

   In this project, instead of using multiple features to train the model, we used only one feature and that is polarity of the sentence. We used TextBlob for extracting polarity and subjectivity as follows. We have following list of features being extracted.

   (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) Third sentence Polarity (1/3, 2/3 and 3/3)
   (g) Third sentence Subjectivity (1/3, 2/3 and 3/3)
   (h) Difference between max and min polarity of the thirds.
   (i) Fourth sentence Polarity (1/4, 2/4, 3/4 and 4/4)
   (j) Fourth sentence Subjectivity (1/4, 2/4, 3/4 and 4/4)
   (k) Difference between max and min polarities of the fourths.

   In this manner,23 features of each sentence was extracted.

   TextBlob is used for extracting the features for input. TextBlob is Python library which is used for performing natural language processing tasks. It performs Natural Language processing functions like noun phrase extraction, part of speech tagging, sentimental analysis, classification (naive Bayes, decision tree), n-gram, etc.

3. Building the model:

   As discussed in the introduction, the project uses a deep neural network, developed using TensorFlow. It was used because of its efficiency and ease of implementation when it comes to deep neural networks. TensorFlow, as the name suggests, allows you to build data flow graphs to perform mathematical computations at every node of the graph. TensorFlow gained so much popularity for deep learning because it provides superior performance as compared to other libraries available. The performance is improved because TensorFlow takes the data flow graph along with the inputs and runs in the background. It brings back the output to the python file. Ultimately avoiding all the performance issues that Python possesses. Apart from these, TensorFlow provides auto-differentiation that helps a lot while implementing gradient-based machine learning algorithms. Aside from that, it has other features which make it the best to implement deep learning models.

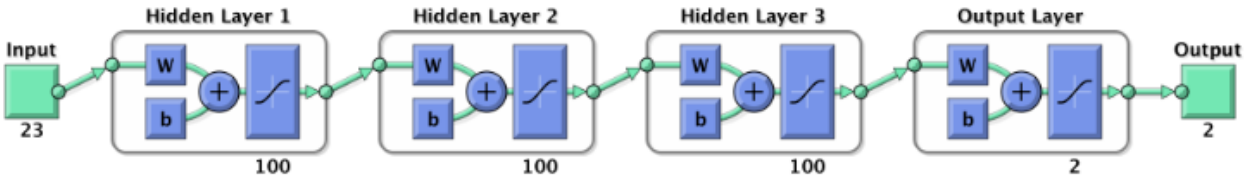

Figure 1: Neural Network Model

A model with three hidden layers and one output layer was built for this project[8]. Each hidden layer has 100 neurons, and output layer has two neurons, one for each class, namely Sarcastic and Regular. One output neuron should be sufficient for a binary classifier, but the model has two so that the output is in the form of one hot encoded array.One hot encoded array is easy to extract and compare results. One hot encoded array is the type of encoding in which the index corresponding to the class is 1 in the array, and the other values are 0.

The first layer receives the weighted features as input, adds all of them, then adds a bias to it and finally passes the whole thing through a logit function[7]. That is implemented in the following lines :

output_layer1= tf.add(tf.matmul(data, hidden_1_layer['weight']), hidden_1_layer['bias'])
output_layer1= tf.nn.relu(output_layer1)

output_layer1 stores the output of layer 1. This output is then passed to layer two as the input, and then there it is added, a bias is added and then it is passed through a logit function as follows:

output_layer2= tf.add(tf.matmul(output_layer1, hidden_2_layer['weight']), hidden_2_layer['bias'])
output_layer2= tf.nn.relu(output_layer2)

Similarly, the output of layer 2 is passed through layer 3, whose output is passed to the output layer, where it is added and then the bias is introduced. Then this value is returned as the output of the neural network.

output = tf.matmul(output_layer2, output_layer['weight']) + output_layer['bias']

The neural network and the data flow is defined. Back-propagation in the network is established to facilitate learning. TensorFlow does this. The output of the neural network for a particular input x is obtained as follows:

prediction = neural_network_model(x)

Then, the cost is computed from the expected output and the predicted output as follows:

cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=prediction, labels=y))

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 as follows:

optimizer = tf.train.AdamOptimizer().minimize(cost)

The goal is defined. The way to achieve the goal is also defined which is stochastic gradient descent. The Adam optimizer is used for this because it provides faster convergence. Aside from that it also chooses a separate learning rate for each attribute. Finally the optimizer and cost are tied together and the neural network is run.

_, c = sess.run([optimizer, cost], feed_dict=x: batch_x, y: batch_y)

The optimizer and cost with the inputs and expected outputs specified in the feed dictionary are run in a TensorFlow session. TensorFlow takes the graph and inputs and separates from normal Python code, does the computation and returns with the cost.

4. Training and Testing:

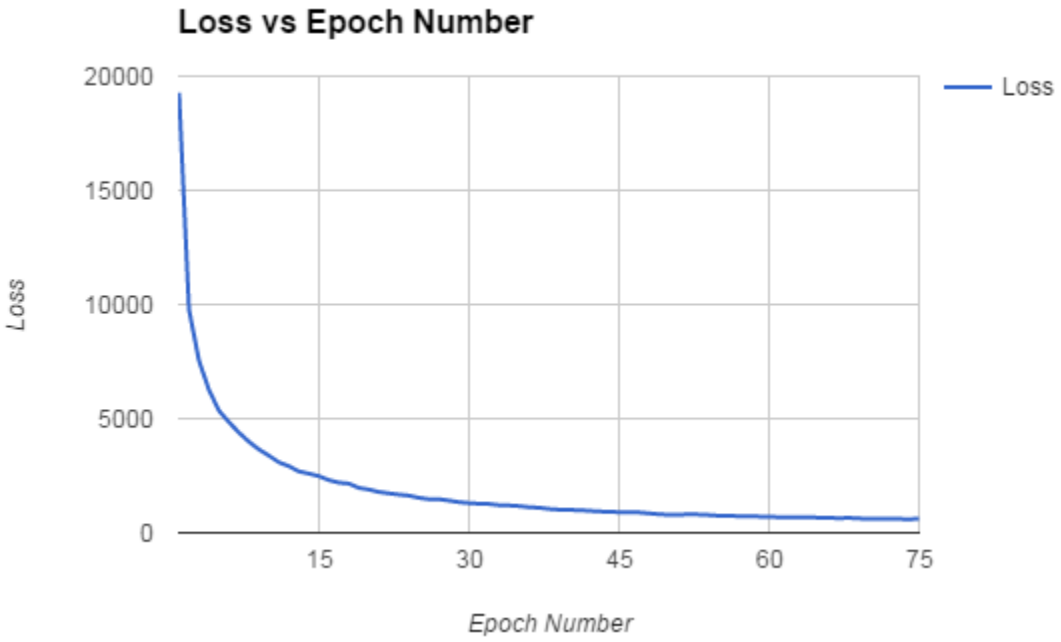

Figure 2: Epoch versus loss

After building the model, it is trained with the extracted features. The training set consists of 70% of the feature set.The testing set consists of the rest 30% of the feature set. The training is done using the Adam Optimizer to minimize the mean cost throughout the epoch.

There are two methods to train a neural network model i.e. online and batch processing. The batch process is done in epochs which are the number of times the entire dataset is processed.For testing the model, the training set should be passed to confirm the predictive power of the neural network.75 epochs of training are done. In each epoch, the batches of 50 are passed.During each epoch, the loss is accumulated. The final epoch loss is printed at the end of the epoch.

From the graph shown in figure 4, it is evident that the loss is reducing after every epoch; which implies that the model is learning in every epoch.

The F1 score or F-measure is a number between zero and one which determines how the neural network works during the testing phase.

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

Precision is the what percentage of sentences that the classifier labeled as sarcastic are actually sarcastic. *precision=true positives / (true positives + false positives*

Recall is the what percentage of sarcastic sentences did the classifier label as sarcastic.

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

5. Visualization on Model: The project used Tensorboard which is provided by TensorFlow to create a visualization of the data flow graph that was built for TensorFlow[1] . All the scalar components which need to be included in the TensorBoard are added using the following command:

   tf.summary.scalar("Scalar name", scalar_variable)

   All the components of the neural network model which are a part of the data flow graph,are included in name scope as follows:

   with tf.name_scope("model"): prediction = neural_network_model(x)

   When all the components are included, then the summary is merged.

   merged_summary_op = tf.summary.merge_all()

   A summary file writer object is created for the TensorFlow to create the log file. TensorFlow collects all the logs into this file during each epoch.This log file is referred to while building the visualization.

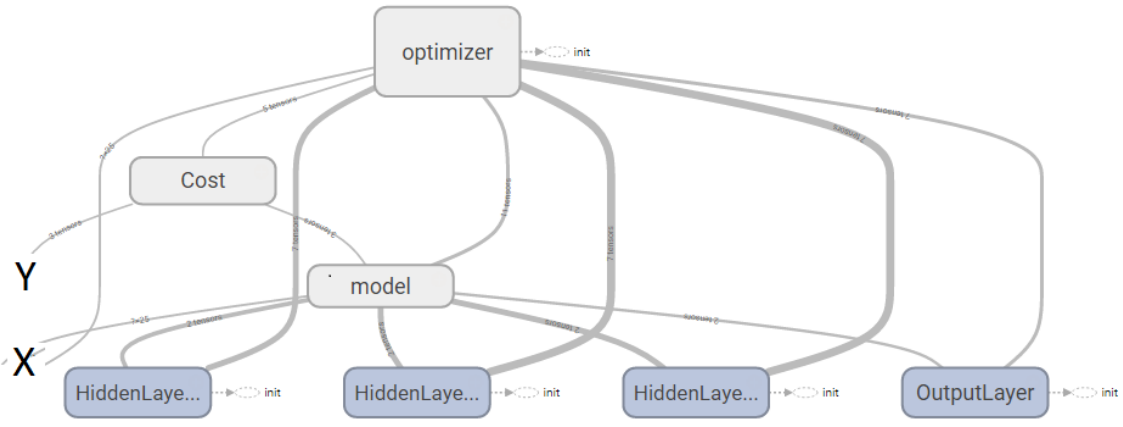

Figure 3: TensorFlow Visualization in TensorBoard

4 Experiment Results and Analysis

The original existing website has an F1 score=.56 for n-gram feature extraction and .41 for sentiments extraction and .36 for topics extraction. The proposed system is expected to have an F1 score of about 0.6, using only one type of feature. The project uses scikit-learn Python package to compute the F1 score .

Sentence	Prediction
I just love the morning alarms!	Sarcastic
Just a really long sentence to check that it is not length that makes the difference.	Regular
I can see your point, but I still think you're full of shit.	Sarcastic
Nice perfume. Must you marinate in it?	Sarcastic
I want to be a doctor someday	Regular
What happened at dinner? My parents spent the money for our wedding. What did you order?	Regular
When I see ads on TV with smiling, happy housewives using a new cleaning product, the only thing I want to buy are the meds they must be on.	Sarcastic

Figure 4: Sample input and output

In the above example, the sentence"What happened at dinner? My parents spent the money for our wedding.What did you order?" is a famous example of a sarcastic sentence from the popular series Friends.This sentence is classified as regular by the project because the sarcasm is based on the context of the conversation.

	Sarcastic	Regular
Sarcastic	3240	4273
Regular	2422	6481

Figure 5: Confusion matrix

From 16416 instances of sarcastic and regular sentences, the model was able to successfully classify 3240 instances of sentences as sarcastic from total of 7513 sarcastic sentences. 6481 instances of regular sentences were classified correctly from 8903.

5 Conclusion

From the observations, it is evident that the project was able to obtain similar performances with the same dataset using TensorFlow with minor features. The performance would improve if more

features were included as the base project.The dataset had very fewer instances. If the model were trained with more instances, then it would result in a better performance. As compared to the base project, F-measure was similar for this project.

References

[1] TensorBoard Visualizing learning , "https://www.tensorflow.org/get_started/summaries_and_tensorboard".

[2] M. Cliche. negproc.npy, "https://github.com/MathieuCliche/Sarcasm_detector/blob/master/app/negproc.npy", August 2014.

[3] M. Cliche. posproc.npy, "https://github.com/MathieuCliche/Sarcasm_detector/blob/master/app/posproc.npy", August 2014.

[4] M. Cliche. The Sarcasm detector, "https://www.github.com/MathieuCliche/Sarcasm_detector", August 2014.

[5] M. Cliche. The Sarcasm detector, "https://www.thesarcasmdetector.com/about", August 2014.

[6] M. Gorner. Intro to Tensor flow and deep learning, "https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist".

[7] H. Kinsley. Intro to deep learning using tensorflow, "http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html".

[8] H. Larochelle. Course content, "http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html".