Sarcasm Detection

Jinal Sapariya, Saloni Somaiya
Email: {sapariya.j,somaiya.s}@husky.neu.edu

I. Abstract

Sarcasm is a form of speech act in which speakers convey their message in form of sharply ironical taunt. The intended meaning differs from semantic understanding of the sentence, syntactic and semantic clues doesnt always help. Automated detection of sarcasm is still in early stages. One reason for lack of computational models has been the difficulty to correctly identify and label sarcastic tweets. We found that a simple Neural Net, with single projection layer meets the performance of general classifiers. In this paper, JSON format file is used as an input for which a machine will be trained. Based on the training, machine will automatically execute its own generated words. We explore the ability of Long short term memory (LSTM) recurrent neural network architecture as compared to normal Recurrent Neural Network. We find out that recurrent neural networks can be used as generative models. This makes a provision that they can be used for predictive models to learn the sequences of a problem and generate new solutions as well. We discover how to create a generative model for text, character by character using Long Short Term recurrent neural networks with Keras and Tensorflow as backend. The main aim of the research conducted in this project is to detect and identify the best classification model to classify the sentences as sarcastic or not. The secondary aim of this project is to find our if Traditional Learning Classifiers outperform neural network classifiers and deep learning networks.

II. Introduction

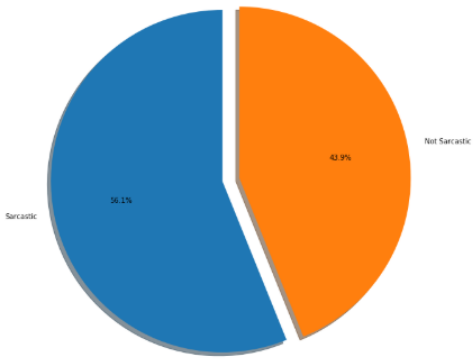
Sarcasm is a form of humor found in many languages that tries to generate humorous content by using words or sentences that may look contradicting to simple English but contains a second meaning for that sentence or a humor effect. It is a speech act where speaker conveys the meaning implicitly. This makes the sarcasm detection task interesting and challenging at the same time. It even hard for humans to sometime detect sarcasm as it is subjective to the con-text, people, language, culture or situation. The main challenge for these problems is to understand and detect sarcasm as it is important to understand the facts related to an event. This allows for detection of contradiction between the objective polarity (usually negative) and the sarcastic characteristics conveyed by the author (usually positive). Consider, the example, I love how you always mess up your cupboard, it is difficult to extract the knowledge needed to detect if there is sarcasm in this statement. In the example, I love provides knowledge of the sentiment expressed by the author (in this case positive), and mess up the cupboard describes a contradicting sentiment (that of negative). However, accuracy and robustness of results are often affected by untruthful sentiments that are of sarcasm nature and this is often left untreated. Past research is mostly focused on either word features or contextual features. In addition, there hasnt been any substantial work on identifying performance of neural nets as way to estimate sarcasm in text. Current social and political scenario demands a better model to identify influential sections in community with lowest form of wit in twitter sphere.

III. Problem Statement

Humor detection is an important problem in the field of Natural Language Processing. Famous voice assistants such as Alexa, google voice assistant, Siri lack the opportunity of detecting sarcasm or humor in general. For example. if we say, Alexa hit me with a truck, it does not understand the sarcasm involved in the sentence. The ability to detect humor is important as it becomes an important feature when it comes to voice assistants and chatbots and we are trying to address the same problem with this project which can help in the field of natural language

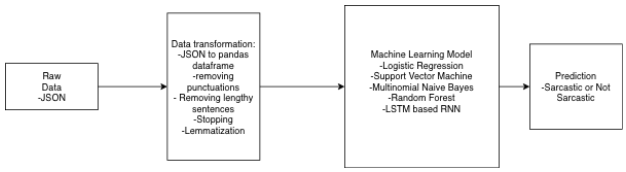
IV. Dataset

We are using News Headlines dataset for Sarcasm Detection which is collected from two news websites. The news headlines are written by professionals in a formal manner, thus there are no spelling mistakes and informal usage. This reduces the sparsity and increases the chance of finding pre-trained embeddings. The news headlines we obtained are self-contained. This would help us in teasing apart the real sarcastic elements. We extracted the data from the dataset consists of links of the articles and headlines provided by Kaggle which was in JSON format. We converted this file into plain text format. The corpus was pretty much clean all we had to do is remove punctuation and case folding on dataset which we extracted. The dataset is of size 26709 of which 14957 are sarcastic sentences and 11,752 are non-sarcastic sentences.



V. METHODOLOGY

Our first approach is to convert the JSON formated data into text data and then clean the given dataset by performing different pre-processing steps and then train the model using various machine learning algorithms.



A. *Data Preprocessing*

The data was provided in JSON format. We converted the data into text format. Then we performed different pre-processing techniques like data cleaning, annotation and no-ramalization

Data Cleaning: In this process we removed less useful parts of the text like digits, punctuation and limiting the length of the sentences. For Example: advancing the world's women Cleaned: advancing the worlds women

As the distribution of the sentences were of the length of size 15-20. we considered all those sentences of the fixed length.

Annotation: Consists of the application of a scheme to text it includes parts of speech tagging.

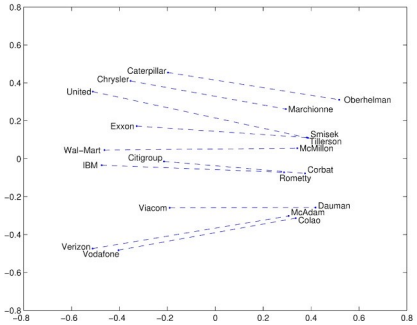
Normalization: This process consists of translation of terms in the scheme reduction Stemming, Lemmazation, Tokenization, Stop word removal and other forms of standardizations. We performed tokenization, lemmatization and stop words were removed.

- Tokenization: Describes the general process of breaking down a text corpus into individual elements that serve as input for various natural language processing algorithms. For Example: new movie taps into nations love Result: new, movie, taps, into, nations, love
- Lemmatization: This process refers to doing things correctly with the use of vocabulary and morphological analysis of words, typically aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.
- Stop Words Removal: This includes removal of irrelevant words. for example: removal of most common words such as 'is', 'the', etc.

B. *2.1 Word Vectorizations*

Word vectorization refers to a set of techniques that aims at extracting information from a text corpus and associating to each one of its word a vector. There are different methods for converting the words into vectors such as CountVectorizer, TF-IDF Vectorizer, Google word2Vec, Gobal word2Vec. We used pre-trained vectors which are generated from GloVe algorithm and CountVectorizer to form the embedding matrix of size 200.

GloVe: GloVe stands for global vectors for word representation. It is an unsupervised learning algorithm developed by Stanford for generating word embeddings by aggregating global word-word co-occurrence matrix from a corpus.



CountVectorizer: Works on Terms Frequency, i.e. counting the occurrences of tokens and building a sparse matrix of documents x tokens.

	Jumps	The	brown	dog	fox	lazy	over	quick	the
Doc1	0	1	1	0	1	0	0	1	0
Doc2	1	0	0	1	0	1	1	0	1

2.2 Embedding Matrix

Word Embedding: Is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. this similarity between the words can be solved building embedding matrix. The following figure shows the format embedding matrix.

hello	12	45	43	26	78	532	...
there	43	25	778	43	53	78	...
texas	34	56	23	12	56	74	...
world	342	54	23	5	7	423	...
...	...						

C. *Machine Learning Models*

Logistic Regression: Logistic regression is used to describe data and to explain the relationship between one dependent variable and one or more nominal, ordinal, interval or ratio-level independent variables. The equation of logistic regression is given by

$$g(z) = \frac{1}{1 + exp(-z)} \tag{1}$$

Cost function:

$$h_\beta(x) = g(\beta^T x) \tag{2}$$

types of logistic regression:

- Binary Logistic Regression
- Multinomial Logistic Regression
- Ordinal Logistic Regression

In our logistic regression based model we use vectorized sentences as features and tries to predict the class of the sentence. The major point to note here is that both train and test corpuses are transformed using a count vectorizer that is fitted on train data.The model uses binary cross entropy as a loss function.

$$H(y) := -\sum_i (y_i^{'} \log(y_i) + (1 - y_i^{'}) \log(1 - y_i)) \tag{3}$$

Support Vector Machine: This model uses a Support Vector Machine (SVM) which is a discriminate classifier formally defined by a separating hyper-plane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyper-plane which categorizes new examples.

More formally, a support vector machine constructs a hyperplane or set of dhyperplanes in a high- or indefinite-dimensional space, which can be used for classification, regression, or other tasks like outlier detection. Intuitively, a good seperation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functoinal margin), since in general the larger the margin, the lower the generalization error of the classifier.

Multinomial Naive Bayes: Naive Bayes is a simple model which works well on text categorization. We use a multinomial Naive Bayes model. Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. This model uses multinomial Naive Bayes linear classifier algorithm. The model uses vectorized sentences and multinomial variants are fitted. Class c is assigned to statement d, where

$$c = argmax_c P_{NB}(c|d) \tag{4}$$

$$P_{NB}(c|d) := \frac{(P(c)\sum_{i=1}^{m} P(f|c)^{n_i(d)})}{P(d)} \tag{5}$$

Random Forest Classifier: A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners.

Random Forest Classifier model constructs a multitude of decision trees at training time and outputting the class that is the mode of the classes. In our experiments we used a random forest 1000 trees to estimate the class of a datapoint.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x') \tag{6}$$

LSTM based RNN: Recurrent Neural Network uses Long Short Term Memory and dense layers to crack down this classification problem.The idea behind RNNs is to make use of sequential information. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output depended on previous computations. The model uses embedded sequences of 200 size as training set and predicts the output with the accuracy of 87% on the validation set.

The Neural Network configuration is described below:

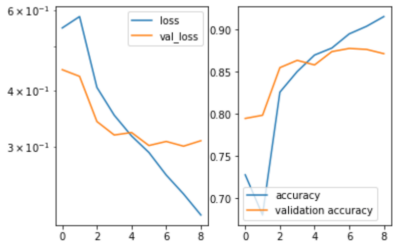


Fig. 1. loss vs. number of epochs

Embedding layer: Transforms words into embedded features and acts as input to hidden layer.

Hidden layer:

1. LSTM with 200 neurons with return sequences i.e. getting the last output feed into current input feed.

2. Final fully connected neural netwrok with one output nodes with sigmoid activation

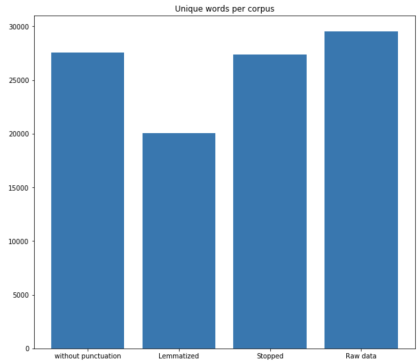
3. Optimizer- adam optimizer of keras

4. Loss function - binary cross entropy epochs - 40. Following are the graphs for training and validation accuracy vs number of epochs and training and validation loss with number of epochs.

VI. Experiments

A. *Experiments on different Natural Language Processing techniques on corpus*

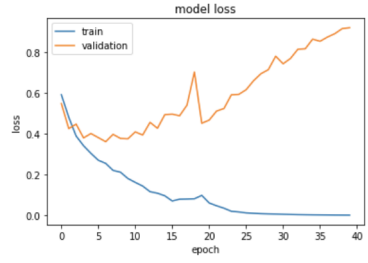
We applied various prepossessing and exploratory data analysis to understand the nature and distribution of data. In this process as mentioned earlier we applied tokenization to experiment a bag of words approach and understand whether or not order matters in actual sentences for our models to detect sarcasm. One of our experiments with input data consisted of using the actual raw data with punctuation and numbers to see what happens and to check whether the punctuation play an important role in deciding sarcasm in a sentence. We also tried lemmatization which checks whether or not the form of the word or just the root of the word matters for detecting sarcasm. We also applied to removal of stop words to our corpus to generate a lemmatized and stopped corpus and various combinations of that to see whether this changes make any difference tot the results and learning process of the model or not. We also tried this to see whether our non Neural Network models i.e. Logistic regression, Random Forest, Naive Bayes which uses count vectorized inputs are actually affected by these changes. Majority of the times we did not see any astronomical difference in the results but some minor changes which helped us understand the nuances and effects of this changes on the data and various machine learning techniques. We fed all our models with this datasets that we created and recorded their accuracy, precision and recall in the process to compare and contrast in the end. Below is the graph for unique words per corpus and we will discuss in evaluation section how it played out for different models.



B. *Experiment of hyperparameter of classification models*

We have five different models in total to compare and contrast their performances on our corpus. We tried these models and tuned the hyperparameters associated with them to get the most accuracy and least loss we can get from the model. For multinomial Naive Bayes we tried to tune alpha parameter by manual trial and error as grid search is only way to tune it, we discovered that model accuracy and loss did not move much and changed in the range of 1e-5 so we decided to keep this value at 0.5 for every corpus. For random forest classification algorithm there are plethora of hyperparameters to tune but we kept majority of them to their default value and tried to tune number of estimators, random state and depth of trees involved in random forest. Surprisingly enough random forest classifier even after extensive tuning did not work well on any of the datasets, which can be attributed to the nature of the data i.e. vectors and not seperated features. For Recurrent Neural Network we did the maximum amount of tuning when compared to our tuning efforts for all other models. First hyperparameter we tried was to tune size of our embeddings and after observing average length and various papers in the field along with our experiments we decided to keep it at 200. Then comes the Dense layers following LSTM layer and we were facing a lot of overfitting in our initial experiments and we observed it was due to the last few Dense layers overfitting the training data. We tried to minimize this with a spatial dropout layer and dropout of 20% after each of the Dense layers. The number of Dense layers is just right according to us because when we observe the training curves for accuracy and losses we can clearly observe less amount of overfitting until model reaches a decent accuracy level.

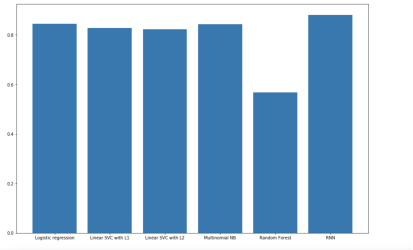
Model overfitting without Dropout



Linear Support Vector Machine was certainly a classification algorithm we wanted to try because it deals with vectors and our data was just that, sentences converted to vectors. We tried out SVC with two major variations, one with L1 regularization and other with L2 regularization, surprisingly enough both of them worked with almost equal scores with every dataset which will be evident in our evaluation section.

VII. Evaluations and Comparisons

Following is the graph with the comparisons of our different models on clean data without punctuations.



As we can see from the above graph that Recurrent neural network performs the best when the dataset is with the punctuations. Also the linear SVC with both L1 and L2 regularization pretty much works similarly. The model without punctuation did not work very well. From our experiments we came to know that punctuations are an integral part of a sarcastic sentence. We also experimented with different models like logistic regression, multinomial naive bayes and random forest. The logistic regression performed very well with the dataset that contained punctuations. The random forest was the worst model and multinomial naive bayes was a moderate model.

In the end we concluded that the models which used count vectorizer as method to vectorize the sentences worked well when we somehow reduced the unique words and the models that depended on embedding worked well when more amount of variability in unique terms were given.Also we noticed this effect with sentences with punctuation gave all over very good accuracy in all methods

VIII. Conclusion

In this project we performed classification on labels to be sarcastic or non-sarcastic using various models. After cleaning and exploration of data we could analyze what pre-processing needs to be done on it. We noticed that the dataset did not contain much of noisy data. We performed various steps in text normalization like removal of punctuations, stop words, digits, tokenization, stemming and lemmetization using parts of speech. We fitted the pre-processed data into different models and trained them using deep learning networks. From evaluation of models we conclude that LSTM/RNN model gave us better accuracy of 86% to correctly classify the labels into sarcastic and non-sarcastic sentences. The poorer results obtained using random forest classifier were not completely expected. We expect to see that some sort of contextual clues to play a big role in sarcastic nature of a sentence. The other classifier's performances were average, this could possibly be due to the small size of the dataset which in turn leads to

the model overfitting the data.In general we applied various Natural Language Processing methodologies to preprocess and classify the text data.

REFERENCES

[1] Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network by Kristina Toutanova, Dan Klein, Christopher Manning, and Yoram Singer.

[2] Automatic sense disambiguation using machine readable dictionaries : How to tell a pine cone from an ice cream cone by M.E. Lesk.

[3] NLTK: The Natural Language Toolkit by Steven Bird.

[4] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[5] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

[6] GloVe: Global Vectors for Word Representation. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014.

[7] Carvalho, P., Sarmento, L., Silva, M.J.: Clues for detecting irony in user-generated contents: oh..!! its so easy;-). In: Proceedings of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion, pp. 5356. ACM (2009)

[8] Liebrecht, C., Kunneman, F., van den Bosch, A.: The perfect solution for detecting sarcasm in tweets not. In: WASSA 2013, p. 29 (2013)

[9] Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

[10] Keras, Chollet, Francois and others,2015

[11] https://www.kaggle.com/

[12] Gibbs Jr, R.W., Colston, H.L.: Irony in language and thought: A cognitive science reader. Psychology Press (2007)