Now That's What I Call Sarcasm: Detecting Sarcasm In Text

Akhila Kotapati[1] Vamsi Krishna Reddicherla[2]

[1]Computer Science
George Mason
University,USA
akotapat@gmu.edu

[2]Data Analytics Engineering
George Mason
University,USA
vreddich@gmu.edu

ABSTRACT

Sarcasm detection is the task of predicting sarcasm in text. Sarcasm is speech or writing where you say something while meaning the opposite. It is ambiguous and misleading, where the accurate comprehension of sentences is hampered. In this paper, we attempt to design machine learning classifiers like support vector machine (SVM) and NaÃŕve Bayes to detect sarcasm in text. We have also proposed a neural network model that is composed Convolutional Neural Network (CNN), Long short term memory (LSTM) network and a Deep Neural Network (DNN). Our result shows that neural networks attained a considerable accuracy improvement compared with traditional machine learning algorithms.

1 INTRODUCTION

Sarcasm is defined as speech or writing which means the opposite of what it seems to say. With the advent of social media and increasing sophistication of language, the usage of sarcasm in conversations has taken a quantum leap altogether. Sarcastic remarks are generally made to mock a person. Sarcastic remarks can both be blunt and harsh, or it can be made in good humor to tease the other person. Detection of sarcasm is a difficult task due to ambiguous nature of sarcasm. The two cues which in general indicate sarcasm in a verbal conversation are speakers tone and the body language of speaker. Humans use these cues to detect sarcasm. The absence of such cues in text increases the difficulty by many folds. The user count of social media platforms has grown by leaps and bounds in the recent years. Users have been using social media to voice their opinion. Many companies have been analyzing data from these social media networks to extract user opinion and user sentiment towards their product. An automatic sentiment analysis tool will interpret a sarcastic comment as positive due to the presence of positive words in

the sentence, such errors will greatly impact the effectiveness of sentiment analysis tools. The inability to detect sarcasm in text also hinders the growth of review summarization applications as a negative review can be interpreted as positive one. The problem of sarcasm detection has attracted many researchers over the years. Majority of the researchers formulated the problem as a sentence classification task. In this paper, we have conducted a rigorous analysis of the current state of the art methods which are being used to detect sarcasm. We have conducted experiments with various possible feature sets and have reported the impact of each of them on sarcasm detection.

2 Related Work

Typically, sarcasm detection was considered as a classification problem. Many researchers handled different aspects of sarcasm problem and explored the use of various features in identifying sarcasm in the text. González-Ibáñez et al. (2011) and Riloff et al (2013) both considered a sarcastic statement as a combination of both positive and negative emotions in a statement. Many experiments are also performed to recognize sarcasm with the use of lexical, syntactic and pragmatic features, and studied the role they play in the detection of sarcasm. In addition to these feature, word-embeddings were also taken as feature, to capture incongruity of context in the absence of sentiment words. Most of the research was done using Twitter data, where the tweets with #sarcasm hashtag provided a supervision for analysis. Many papers have also proposed different neural network architectures for sentiment analysis, one analysis is done using convolutional neural network (CNN) with max pooling, the other is done as a combination of both convolutional neural network (CNN) and Deep Neural Networks (DNN) for sarcasm detection. Most of the experiments using neural networks gave a highly reasonable results. The major advantage of using neural networks is that even subtle semantic factors that are difficult to capture using manual features are discovered.

3 Features

N-grams: Individual token, bigrams and trigrams are taken from the data. Through this we can draw similarities and can extract their sentiment.

TF-IDF: Normalizing frequencies can assign more weightage to the important words. This is useful in further extraction of features.

Punctuation-Based features: We took all the punctuations in the comments. A comment with more than one exclamatory or question mark may have high chance of it being a sarcastic statement. So, we tried to take advantage of this, in addition to the number of words in a comment.

Sentiment and POS: Extracting sentiment can be helpful when detecting sarcasm as a sarcastic statement can be considered as a combination of both negative and positive sentiment. POS tags can be helpful in predicting number of nouns, verbs and adjectives in the comment. If a person uses more number of adjectives, then there is a high chance that he is being sarcastic.

Word-embeddings: Word embeddings are

highly successful in predicting the context of a given word and also models like GloVe give the count of frequent words in the form of co-occurrence matrix which is helpful in predicting the context with respect to the overall corpus.

4 Approach

We have formulated the problem of sarcasm detection as a classification task. We will be using supervised machine learning and deep learning algorithms to classify sentences as sarcastic or not - sarcastic. Our main aim of the project was to understand the effect of features which are currently being used for sarcasm detection. We also wanted to come up with the best possible combination of features and algorithms to achieve high accuracy. We have come up with a guide which researchers can refer to before starting their project.

4.1 Naïve Bayes

Naïve Bayes is a simple classifier that calculates the probability of each category and outputs the category with highest probability. We created a multinomial naïve bayes, as it suitable for the classification of data with discrete features. Problem instances that are represented as vectors of feature values are given a class label by the model.

$$P(\theta|\mathbf{D}) = P(\theta)\frac{P(\mathbf{D}|\theta)}{P(\mathbf{D})} \quad ||I, \tag{4.1}$$

4.2 Support Vector Machine

Support vector machine has been a popular machine learning algorithm for natural language processing tasks. SVM classifier learns a classification hyperplane so that the nearest point distance is maximized. Through this, it linearly separates the data in high and sparse feature space, which is very helpful for NLP tasks. It may be possible that the data id non-linear, in such cases, SVM uses the kernel trick. We can use a Gaussian kernel in order to map the data into feature space. In addition to the penalty parameter C, Gaussian kernel has gamma parameter that interprets the influence of a single training example. In Python library scikit-learn, we have used SVC function. We have used different set of features to learn the performance of support vector machine.

4.3 Feed Forward Neural Network

Our Baseline method in deep learning was the basic Feed Forward Neural Network. The Feed forward neural nets are characterized by a combination of layers and probabilistic functions for more complex problems which cannot be solved by linear classifiers. These typically consist of three kind of layers which are input, hidden, output layers. These layers are connected with each other by a set of weights which are learned using an algorithm called Backpropagation. We used RMS prop(a variant of gradient descent) to train our network.

4.4 Long Short Term Memory

Long short term memory network is a variant of recurrent neural network which are capable of learning long term dependencies. An LSTM unit is composed of three gates: input i, forget f, output o and two memory cells: existing memory cell, c, and new memory cell content, ËIJc. The main idea is that by using gates to control the flow of information in the unit, it can decide whether to keep or discard certain information in the memory cells for each layer at time step t.

The transitional functions are as follows:

$i_t = \sigma(W_i \,.\, [h_{t-1}, x_t] + b_i$

$f_t = \sigma(W_f \,.\, [h_{t-1}, x_t] + b_f$

$\acute{C}_t = \tanh(W_C \,.\, [h_{t-1}, x_t] + b_C)$

$C_t = f_t \bullet C_{t-1} + i_t \bullet \acute{C}_t$

$o_t = \sigma(W_o \,.\, [h_{t-1}, x_t] + b_o$

$h_t = o_t \bullet \tanh(C_t)$

4.5 Gated Recurrent Unit

Gated Recurrent unit is another variant of recurrent neural networks which have been introduced recently. The architecture of both LSTM unit and GRU are similar to each other. The only difference between the traditional LSTM and GRU is that the GRU doesn't have an output gate. This leads to exposure of memory at each time step. GRU works as well as LSTM in many tasks.

5 Experiments

5.1 Dataset

We have taken our data from the self annotated reddit corpus (SARC), which is a large dataset of over 1.3 million sarcastic statements. Reddit data ensures that every user includes the symbol \ s whenever they post a sarcastic comment, and thus better than the one that is manually annotated. In our project we considered "politics" subreddit data. This dataset is particularly balanced, that is they have equal number of sarcastic and non sarcastic statements.

5.2 Preprocessing

We have used some data preprocessing techniques to remove noisy data, stop words etc. Data like punctuations, special characters, and numeric, which are not helpful in training are also removed using techniques like punctuation marks removal, special characters removal and numeric removal. Then, converted all the words into lowercase, followed by tokenization.

5.3 Naïve Bayes

We constructed a multinomial naïve bayes classifier to train the data. A term frequency-inverse document frequency (TF-IDF) was created using the preprocessed data. In addition to that, POS tag and punctuation based features were created and all these features are fed into the multinomial naïve bayes classifier.

5.4 Support Vector Machine

We have experimented on both linear and non-linear support vector machines. First we trained the linear SVM with several features like word2vec, sentiment, punctuations and tf-idf vectors. Later, trained the data using non-linear SVM, as the data may not be completely separable in linear SVM. Several experiments were conducted on non-linear SVM classifier with different features, where one-leave out approach was implemented in which we observed similar results that implies all features are equally important. In order to optimize the algorithm, we have experimented with few of the random C and gamma parameters. This is implemented by performing grid-search on the classifier.

5.5 Feed Forward Network

The Feed forward neural networks was our baseline deep learning model. We experimented with all the features we had at our disposal. We started with lexical features like n-grams and char- ngrams and tf-idf counts then followed them up with the rest of the features such as Sentiment, Pos-based Features and Topics. We built a simple two layer neural network with 100 input nodes in the first layer and 5 hidden nodes in the second layer. We ran multiple evaluations of our algorithm to find the optimal hyperparameters for each feature subset.

5.6 Convolutional Neural Network

We used the CNN based architecture which used for sentence classification to classify sarcastic sentences. We modified the architecture to optimize it ability to classify sarcastic sentences. The inputs to the architecture were Word2vec and Glove embeddings.

The model has 3 convolution layers at the input. The embedding layer is connected to all three convolution layers. The outputs passed through maxpooling layers before they are merged. The merged output is flattened before it pushed into the dense layer. Relu activation function is used as the acttivation in the hidden layers and sigmoid activation function is used at output layer.The hyperparameters such as filter size ,stride were optimized by trial and error.

5.7 CNN-LSTM-DNN

The architecture of CNN-LSTM - DNN has been the source of many of the exciting applications in Natural language processing. The feature learning ability of CNNs coupled with the sequence processing ability has lead to many break throughs in the field of machine translation. In this paper we have replaced the LSTM unit in this architecture with a bidirectional GRU unit. We believe that by using a bidirectional GRU the application will be able to better retain to context of the sentence. We used GRU instead of an LSTM as the training time of GRU lower than the that of LSTM.

Table 6.1: Naive Bayes

	Precision	Recall	F1 Score
Not Sarcastic	0.58	0.72	0.64
Sarcastic	0.63	0.47	0.54
avg	0.60	0.59	0.59

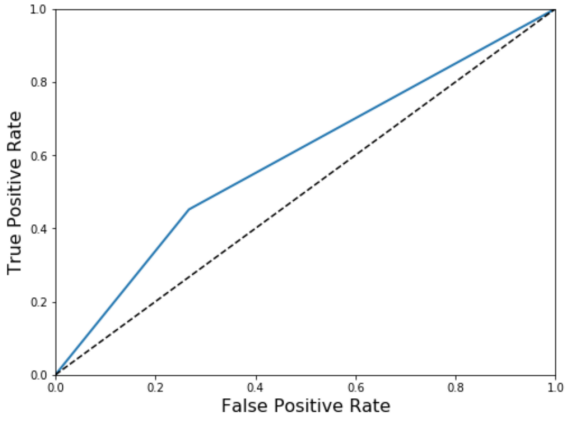
Table 6.2: Grid-Search best parameters

'C' value	Kernel	gamma	Mean accuracy
0.1	linear	-	0.684
0.5	rbf	0.1	0.677
150	rbf	0.0001	0.678
500	rbf	0.0001	0.699

6 Results

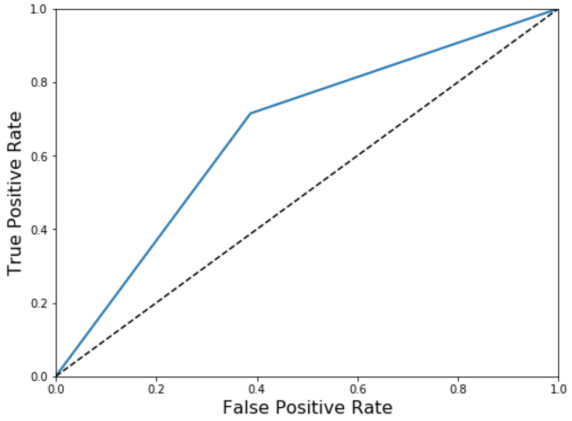
6.1 Naive Bayes

Features that are fed into naïve bayes classifier are punctuation based features and tf-idf vectors. Based on the training data we have generated the confusion matrix to visualize the accuracy of the model. From table 5.1, we can observe that the precision, recall and F1 score varied for both sarcastic and not sarcastic classes. Precision value in the table shows that there few false positive, whereas the recall shows that naive bayes classified more false negatives.



6.2 Support Vector Machine

For support vector machine, we have performed grid search to evaluate the best parameters for the given model. As you can see in the table, for a low value of C, the linear SVM yields a good result. And for non-linear SVM, better results are obtained as we increase the value of C and decrease gamma.



6.3 Neural Networks

We conducted our experiments on a balanced data set. The F-1 Scores compared to current state of the art models are on the lower side, but that does not tell us the full story. Results on this particular dataset have been hard to come by. Using a small subset of data instead of the whole corpus of comments has dearly effected the outcome of the project. Optimization of hyperparameters was one core challenges of this project. We tested our models with different hyperparameter settings for each model.

Table 6.3: ML Classifiers

Model	Features	Precision	Recall	F1
Naive Bayes	TFS+Punctuation+N-gram	0.66	0.42	0.54
SVM	Punctuation	0.71	0.24	0.36
SVM	Sentiment	0.60	0.32	0.43
SVM	Word2Vec	0.52	0.71	0.60
SVM	S+P+W+N	0.70	0.67	0.69

Overfitting was a major challenge we were facing during the evaluation time of the models. Regularization techniques like dropout, batch normalization worked well for the simpler feed forward network but failed in case of larger architectures. We had to augment data to avoid overfitting of models during training. We evaluated the performance of LSTM by repeatedly changing the value of HMU. We also repeatedly changed the values of dropout , Learning rate and weight decay to optimize the accuracy of the algorithm. We also observed that scaling the numeric inputs before passing them into neural network will increase the accuracy by a substantial amount.

7 Conclusion

In this paper we performed experiments using the features which have been invented in past works. We experimented mainly with four kind of features. We can also safely say that we will get rich dividends if we try to solve the problem of sarcasm detection using deep learning. One of the major flaws of our project was the choice of dataset. The lack of computational power to process the larger dataset was another major hinderance to our progress. We trained our models using an AWS EC 2 p2 x large instance with GPU power. In the Future, we would like to explore behavioral approach for sarcasm detection and also would like to get our hands on a larger dataset.

8 Contributions

- Akhila Kotapati
  - Data Acquisition, Data Preprocessing, Word2Vec,LDA
  - Naive Bayes, Support Vector Machine.
  - Tools: Tableau, Jupyter.
- Vamsi Krishna Reddicherla
  - Data Acquisition, Data Preprocessing, LDA (Topics over time),GloVe,LSA.
  - CNN, LSTM, GRU, DNN.

References

[1] Nick Guo, Ruchir Shah, *Finding sarcasm in reddit postings:A deep learning Approach.*

Table 6.4: NN Result

Model	**Feature/Hyperparameter**	**F-1**
DNN	BOW	0.65
DNN	Numerical Features	0.62
DNN	LSA Embeddings	0.48
DNN	LDA Embeddings	0.46
CNN based	Filter size = 128, Filter width = 3	0.68
CNN based	Filter size =64, Filter width = 3	0.68
CNN + Bi(GRU)	char n-grams, Filter size = 64, Filter width = 3	0.72
CNN + Bi(GRU)	Word2vec embeddings,Filter size = 128, HMU = 256	0.70

[2] Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, Ruihong Huang. *Sarcasm as Contrast between a Positive Sentiment and Negative Situation* . 26 (EMNLP 2013).

[3] Roberto GonzÃąlez-IbÃąÃśez, Smaranda Muresan, Nina Wacholder. *Identifying Sarcasm in Twitter: A Closer Look*

[4] Aniruddha Ghosh,Tony Veale. *Fracking Sarcasm using Neural Network).*

[5] Chun-Che Peng, Mohammad Lakis , Jan Wei Pan. *Detecting Sarcasm in Text: An Obvious Solution to a Trivial Problem.*

[6] Aditya Joshi ,Vaibhav Tripathi, Kevin Patel, Pushpak Bhattacharyya, Mark Carman. *Are Word Embedding-based Features Useful for Sarcasm Detection?*

[7] Aditya Joshi,Pushpak Bhattacharyya, Mark Carman. *Automatic Sarcasm Detection: A Survey.*

[8] Debanjan Ghosh, Weiwei Guo, Smaranda Muresan. *Sarcastic or Not: Word Embeddings to Predict the Literal or Sarcastic Meaning of Words.*

[9] David Bamman and Noah A. Smith. *Contextualized Sarcasm Detection on Twitter*

[10] Meishan Zhang , Yue Zhang, Guohong Fu. *Tweet Sarcasm Detection Using Deep Neural Network.*

[11] X. Wang and A. McCallum. *Harnessing Cognitive Features for Sarcasm Detection* . Abhijit Mishra , Diptesh KanojiaâĂă, Seema Nagar , Kuntal Dey , Pushpak Bhattacharyya.

[12] Lakshya Kumar, Arpan Somani, Pushpak Bhattacharyya. *âĂIJHaving 2 hours to write a paper is fun!âĂİ: Detecting Sarcasm in Numerical Portions of Text*