

Sentiment Analysis - Sarcasm Detection in Twitter

Athira M R¹, Chithra C², Gayathry Anil³ and Smitha E S⁴

^{1,2,3}Computer Science and Engineering (Pursuing), Dept. of Computer Science and Engineering, LBS Institute of Technology of Technology for Women, Thiruvananthapuram, Kerala, India.

⁴Associate Professor, Dept. of Computer Science and Engineering, LBS Institute of Technology for Women, Thiruvananthapuram, Kerala, India.

Abstract: Sarcasm is the way of expressing opposite of what exactly is given. Generally, people use sarcasm to criticize. Sarcasm is merely a synonym of irony. Sarcasm is prevalent everywhere including social media. Twitter is a micro-blogging platform extensively used by people to express thoughts, reviews. It is a new trend to post sarcastic statements in order to avoid direct negativity. Hence it is necessary to implement an automated system that will be capable of understanding whether a statement is sarcastic or not. Already there are some systems that are implemented to meet this goal. The aim here is to develop a simpler model for finding the sarcasm in statements using NLP and an algorithm which produces results of good accuracy.

Keywords: Machine Learning, Natural Language Processing, Preprocessing, Sarcasm Detection, Sentiment Analysis, Support Vector Machine

Date of Submission: 01-08-2020

Date of Acceptance: 16-08-2020

I. Introduction

Sarcasm is defined as a cutting, often ironic remark intended to precise contempt or ridicule. Sarcasm detection is that the task of correctly labeling the text as 'sarcastic' or 'non-sarcastic'. It is a challenging task owing to the lack of intonation and facial expressions in text. Recognizing sarcasm in text is a crucial task for tongue processing to avoid misinterpretation of sarcastic statements as literal statements. Accuracy and robustness of NLP models are often affected by untruthful sentiments that are often of sarcastic nature. Thus, it is important to filter out noisy data from the training data inputs for various NLP related tasks. The use of sarcasm is prevalent across all social media, micro-blogging and e-commerce platforms. Sarcasm detection is imperative for accurate sentiment analysis and opinion mining. Twitter is a micro-blogging platform extensively used by people to express thoughts, reviews, discussions on current events and convey information in the form of short texts. We have proposed an algorithm to understand a sentence or tweet is sarcastic or not. This will result in obtaining accurate sentiment analysis and opinion mining. It could contribute to enhanced automated feedback systems within the context of customer-based sites.

II. Related Work

The growth of social media has been exponential in the recent years. Immense amount of data is being put out onto the public domain through social media. When it comes to understanding the meaning of data in social media, Sentiment analysis, the task that involves building a system that categorizes the text data, is an important topic that was discussed by Anukarsh G Prasad et al[1]. They compared various classification algorithms such as Random Forest, Gradient Boosting, Decision Tree, Adaptive Boost, Logistic Regression and Gaussian Naïve Bayes to detect sarcasm in tweets from the Twitter StreamingAPI. By using opinion mining, Sana Parveen et al[2] shows a work that identifies sarcasm in tweets and enhances sentiment analysis. Automatic sarcasm detection, one of the hardest challenges in Sentiment Analysis, was discussed by Paras Dharwal et al[3]. Sindhu. C et al[4]discussed various methodology and techniques used in sarcastic text detection for Sentiment Analysis. Many feature extraction techniques were implemented. Several classifiers were used in various researches such as Support Vector Machine (SVM), Naïve Bayes, AdaBoost, Random Forest etc. A pragmatic classifier that detects emoticon based sarcasm was proposed by Tanya Jain et al[5]. A system that can detect sarcastic tweets with and without context incongruity was discussed by Sreelakshmi K et al[6]. Support Vector Machines (SVM) and Decision Tree were used for modeling the proposed system and both obtained promising results. Manoj Y. Manohar et al[7] proposed a NLP and CORPUS based approach to detect sarcasm on Twitter. They compared the data with the ontology based emotion detection and classify the tweets as a Sarcastic or non-Sarcastic and emphasized the importance NLP and CORPUS based for the detection of sarcastic statements.

Presently there are many systems for detecting sarcasm in twitter data. The most prominent and efficient system proposed uses recurrent neural networks for classification. This system presents a rule-based approach to detect sarcasm expressed due to numbers. This approach compares numerical magnitudes with those seen in similar contexts during a training dataset. Since 'similar context' is key here, the system considers two variants of our approach in order to match the context. Further it proposes deep learning - based approaches to numerical sarcasm detection on social media that doesn't require extensive manual feature engineering. It develops a Long-short Term Memory (LSTM) network which can handle sequences of any length and capture long term dependencies. Most of the existing systems provide a system with an accuracy around 70-75%. Here we are trying to implement a simple system with better accuracy than existing systems.

III. Methodology

Following methodology was adopted to perform Sarcasm detection in tweets

1. **Acquiring Dataset:** The dataset chosen is a file containing nearly 26,000 tweets. The dataset has two columns; type and text. The first column specifies the type of the tweet, i.e., sarcastic or non-sarcastic. Sarcastic tweets are represented by the value 1 and non-sarcastic tweets are represented by the value 0. The second column specifies the tweet. This same dataset is used for training and testing the model. The vector id of text is stored in variable X and value of type is stored in Y. Data in X is divided into X-train and X-test. Corresponding Y is divided as Y-train and Y-test.

2. **Data Preprocessing:** NLP is a branch of data science that consists of systematic processes for analyzing, understanding, and deriving information from the text data in a smart and efficient manner. By utilizing NLP and its components, one can organize the massive chunks of text data, perform numerous automated tasks and solve a wide range of problems such as – automatic summarization, machine translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition, and topic segmentation etc. Here the RE package of python performs the following tasks: Removal of new lines and tabs, Removal of punctuations, separating hash tagged words, Tokenizing the inputs.

3. **Vectorization:** In very simplistic terms, vectorization or word embedding is the process in which the texts are converted into numbers and there may be different numerical representations of the same text. The different types of word embedding can be broadly classified into two categories-

- a.) Frequency based Embedding
- b.) Prediction based Embedding

We have used Prediction based Embedding to map a word using a dictionary to a vector. One of the prediction-based techniques is the CBOW technique which is used to create our word vector.

Continuous Bag of Words(CBOW)[8]: The way CBOW works is that it tends to predict the probability of a word given a context. A context may be a single word or a group of words. The CBOW model architecture tries to predict the current target word (the center word) based on the source context words (surrounding words). For SVM classifier to work efficiently, the input text must be converted to vectors, thus making vectorization an important step in the process. For each of the words in the sentence it creates a one-hot encoded vector.

Word2vec represents words in vector space representation. Word2vec is a two-layer network where there is input, one hidden layer and output. CountVectorizer is the module which is used to store the vocabulary based on fitting the words in it. This is imported from the sklearn. Apply a bag of word approach to count words in the data using vocabulary. If word or token is not available in the vocabulary, then such index position is set to zero. This is converted to an array. This will provide the count of each token in the sentence or list.

4. **Dataset Splitting:** Here the same dataset is used for training and testing. 20% of the dataset is given for testing and the remaining 80% is used to train the model to obtain the output.

5. **Data Classification:** Machine learning is a subset of artificial intelligence that provides computers with the ability to learn without being explicitly programmed. Here the prediction problem is a supervised learning problem, where we must infer from historical data the possible nonlinear dependence between the input (past embedding vector) and the output (future value). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide

as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

In the case of support-vector machines[9], a data point is viewed as a n -dimensional vector (a list of numbers), and one can separate such points with a n -dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. SVMs are helpful in text and hypertext categorization.

Linear SVM [9]: Given a training dataset of points of the form $(x_1, y_1), \dots, (x_n, y_n)$, where y is either 1 or -1 , each indicating the class to which the point x belongs. Each x is a n -dimensional real vector. Then find the "maximum-margin hyperplane" that divides the group of points for which $y=1$ from the group of points for which $y=-1$, which is defined so that the distance between the hyperplane and the nearest point x from either group is maximized.

Any hyperplane can be written as the set of points 'x' satisfying; $w \cdot x - b = 0$ where 'w' is the (not necessarily normalized) normal vector to the hyperplane. This is much like Hesse normal form, except that 'w' is not necessarily a unit vector. The parameter $b/|w|$ determines the offset of the hyperplane from the origin along the normal vector 'w'.

6. **Result Prediction:** The model will predict 0 if the tweet is non sarcastic in nature. It will predict 1 if the tweet is sarcastic in nature.

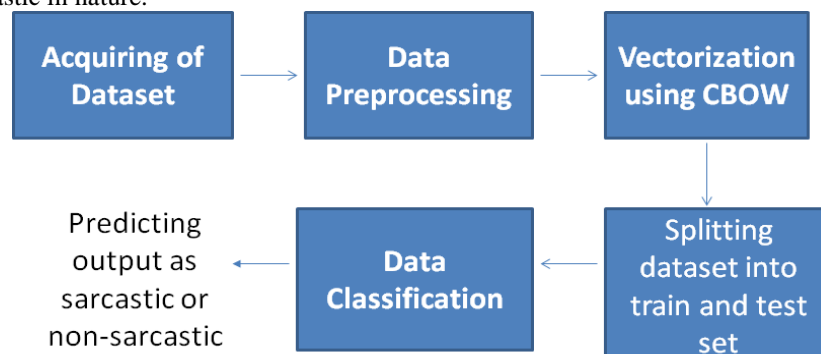


Figure 1: System architecture

Proposed Work

Here a simple machine learning algorithm Support Vector Machine is used to classify whether the data is sarcastic or not. An SVM model may be a representation of the examples as points in space, mapped in order that the samples of the separate categories are divided by a transparent gap that is as wide as possible. New examples are then mapped into that very same space and predicted to belong to a category supporting the side of the gap on which they fall.

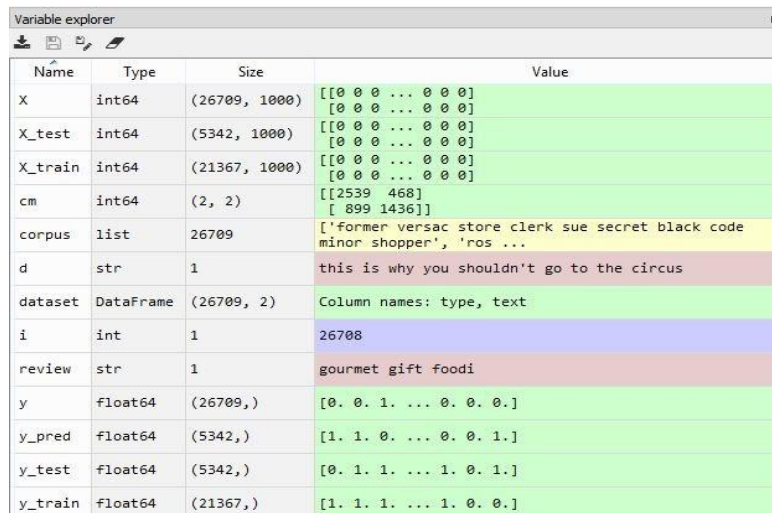
The steps involved are

1. Using the RE package in natural language toolkit (NLTK), preprocessing of the dataset is done. It involves stemming, removing stop words, irrelevant data etc.
2. Vectorizing the preprocessed dataset to obtain the corresponding vector id. CBOW is the technique used.
3. Vector ids of datasets are stored in the X in the form of an array and corresponding value which is sarcastic or non-sarcastic is stored in Y.
4. Splitting the datasets into training sets and test sets. 20 % of the dataset is taken for testing.
5. Providing X-train and Y-train to the classifier and fitting them to SVM.
6. Predicting the test results using X-test.
7. Output is either sarcastic or non-sarcastic.

The entire project is done in Spyder using python language. It provides project support allowing work on multiple development efforts simultaneously. It helps in having step by step execution.

IV. Results

After running the source code in Spyder, we obtain an intermediate result that shows the confusion matrix generated for the input dataset. The system classifies, detects the sentiments and gives the result either as 'sarcasm' or 'non sarcasm'. Figure 2 shows the confusion matrix obtained when the dataset was given to our proposed model.



Name	Type	Size	Value
X	int64	(26709, 1000)	[[0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]
X_test	int64	(5342, 1000)	[[0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]
X_train	int64	(21367, 1000)	[[0 0 0 ... 0 0 0] [0 0 0 ... 0 0 0]
cm	int64	(2, 2)	[[2539 468] [899 1436]]
corpus	list	26709	['former versac store clerk sue secret black code minor shopper', 'ros ...
d	str	1	this is why you shouldn't go to the circus
dataset	DataFrame	(26709, 2)	Column names: type, text
i	int	1	26708
review	str	1	gourmet gift foodi
y	float64	(26709,)	[0. 0. 1. ... 0. 0. 0.]
y_pred	float64	(5342,)	[1. 1. 0. ... 0. 0. 1.]
y_test	float64	(5342,)	[0. 1. 1. ... 1. 0. 1.]
y_train	float64	(21367,)	[1. 1. 1. ... 1. 0. 0.]

Figure 2 Results showing the Confusion Matrix for given dataset

A confusion matrix is a table that is often used to describe the performance of a classification model (or “classifier”) on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm

Figure 3 shows the measures that can be calculated from values in the confusion matrix

Specificity	0.7542	$SPC = TN / (FP + TN)$
Precision	0.8444	$PPV = TP / (TP + FP)$
Negative Predictive Value	0.6150	$NPV = TN / (TN + FN)$
False Positive Rate	0.2458	$FPR = FP / (FP + TN)$
False Discovery Rate	0.1556	$FDR = FP / (FP + TP)$
False Negative Rate	0.2615	$FNR = FN / (FN + TP)$
Accuracy	0.7441	$ACC = (TP + TN) / (P + N)$
F1 Score	0.7879	$F1 = 2TP / (2TP + FP + FN)$

Figure 3: Measures calculated from confusion matrix

V. Conclusion and Discussion

Most of the existing models use RNN to detect sarcasm and then predict whether a tweet is sarcastic in nature or not. For a dataset containing around 40,000 tweets, this model gives a resultant accuracy of 75%. They usually do not support hashtag removal.

In our work, a dataset of 26000 tweets is used for training and obtained an accuracy of 75%. Several datasets of different sizes were used to check the efficiency of this model. The preprocessing of raw data is done using NLP. Then a dictionary of the words is created. This dictionary is used as a reference to vectorize the data using the CBOW method. 20% of the dataset is given for testing and the remaining 80% is given for training. The analysis is that using a simple algorithm we could obtain an accuracy of 75% for a dataset of 26k tweets which supports hashtag removal. Also found out a fact that, in this model accuracy increases with the size of the dataset chosen.

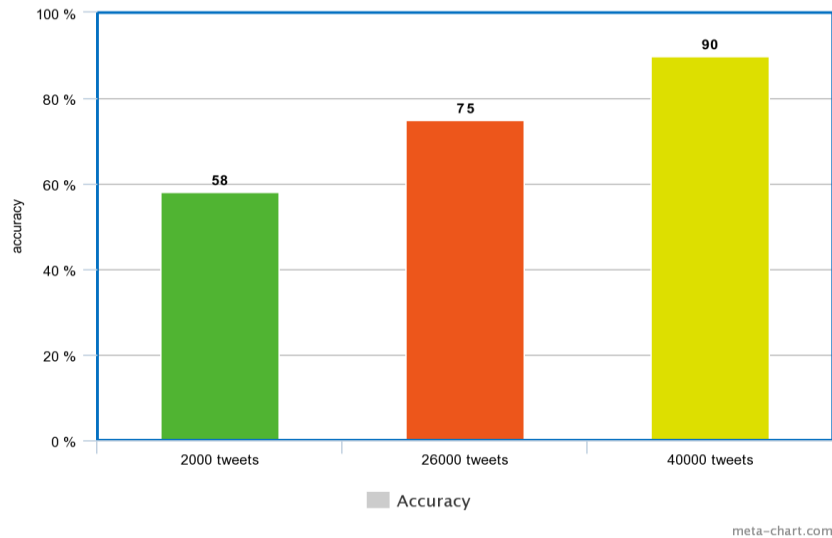


Figure 4: Comparing accuracies of datasets of different sizes

References

- [1]. Anukarsh G Prasad, Sanjana S, Skanda M Bhat, B S Harish "Sentiment Analysis for Sarcasm Detection on Streaming Short Text Data", 2nd International Conference on Knowledge Engineering and Applications, IEEE, 2017.
- [2]. Sana Parveen, Sachin N. Deshmukh, "Opinion Mining in Twitter – Sarcasm Detection" International Research Journal of Engineering and Technology (IRJET), volume 04, issue 10, pages 201-204, October 2017.
- [3]. Paras Dharwal, Tanupriya Choudary, Rajat Mittal, Praveen Kumar, "Automatic Sarcasm Detection using Feature Selection", International Conference on Applied and Theoretical Computing and Communication Technology, IEEE, 2017.
- [4]. Sindhu. C, G. Vaidhu, Mandala Vishal Rao, "A Comprehensive Study on Sarcasm Detection Techniques in Sentiment Analysis", International Journal of Pure and Applied Mathematics, volume 118, pages 433-442, 2018.
- [5]. Tanya Jain, NileshAgrawal, GarimaGoyal, Niyati Aggrawal, "Sarcasm Detection of Tweets: A Comparative Study", Tenth International Conference on Contemporary Computing (IC3), IEEE, August 2017.
- [6]. Sreelakshmi K, Rafeeqe P C, "An Effective Approach for Detection of Sarcasm in Tweets", International CET Conference on Control, Communication, and Computing (IC4), pages 377-382, IEEE, July 05-07, 2018.
- [7]. Manoj Y. Manohar, PallaviKulkarni, "Improvement Sarcasm Analysis using NLP and Corpus based Approach", International Conference on Intelligence Computing and Control Systems (ICICCS), IEEE, 2017.
- [8]. <https://en.wikipedia.org/wiki/Word2vec>
- [9]. https://en.wikipedia.org/wiki/Support_vector_machine

Athira M R, et. al. "Sentiment Analysis - Sarcasm Detection in Twitter." *IOSR Journal of Computer Engineering (IOSR-JCE)*, 22(4), 2020, pp. 42-46.