Hybrid-Model for Sarcasm Detection in a Text

Nimisha Katyayani
PES1201700083
PES University
nimisha.katyayani@gmail.com

Prathiba P.
PES1201700112
PES University
pprathiba.1999@gmail.com

Ashwini M. Joshi
Faculty, Computer Science
PES University
ashwinimjoshi@pes.edu

Abstract - *Sentiment Analysis is the process of determining whether a piece of writing is positive, negative or neutral. A sentiment analysis system for text analysis combines natural language processing (NLP) and machine learning techniques to assign weighted sentiment scores to the entities, topics, themes and categories within a sentence or phrase. The work done here is the detection of sarcasm, in a given text, as entered by the user. Multiple machine learning models have been trained here, finally building a hybrid model, and thus obtaining a classifier, which predicts whether a text is sarcastic or not, at a very high accuracy rate. The dataset is obtained from www.kaggle.com and consists of three columns, the article link, the headline, and a column which describes whether or not the headline under analysis is sarcastic or not, with 0 being non-sarcastic and 1 being sarcastic. The proposed work here first starts with the K-nearest neighbors model, which takes as input the original dataset, along with a number of neighbors. The dataset is split into training and test, and then the model is trained on the training set. Once training is complete, the KNN model is used to make predictions about the test dataset, and thus a new set of target labels consisting of 0s and 1s is obtained. The work here replaces the original target labels with these predicted labels and thus a new dataset is obtained with the original training set, combined with the testing set, with the predicted target labels. The same procedure is followed in each of the classifiers used following KNN, that is, the support vector machine classifier, followed by Naive Bayes Classifier and then Logistic Regression Classifier. At last, through this work, a dataset is obtained, which consists of quite highly accurate values for the column, is_sarcastic, for all the headlines in the dataset, thus building a hybrid model to construct an accurate dataset. In the work as given here, the hybridisation is applied in the way the dataset is constructed, and not on the classifiers themselves.*

Keywords - *Opinion Mining, Sarcasm, Hybrid, Classification*

1. Introduction

Sentiment analysis on natural language has been one of the most targeted research topics in NLP in the past decade, as shown in several recent surveys. Since the goal of sentiment analysis is to automatically detect the polarity of a document, misinterpreting irony and sarcasm represents a big challenge. Sentiment analysis is the interpretation and classification of emotions (positive, negative and neutral) within text data using text analysis techniques. Sentiment analysis allows businesses to identify customer sentiment toward products, brands or services in online conversations and feedback.

There are different types of Sentiment Analysis, for example, Multilingual Sentiment Analysis, Aspect Based Sentiment Analysis, Emoticon/Emojis Sentiments Analysis, and so on. Sarcasm detection in natural language is one of the many variations of sentiment analysis. Sarcasm is one of the main challenges as faced by a programmer, when constructing a model for sentiment analysis, as it is hard to interpret due to the various disambiguities in terms of the words used. A sentence which has a hint of sarcasm to it, uses positive words to describe a negative situation, which makes it hard to detect, as when mining the sentiments from text, these words are just considered to have a positive sentiment, and thus the sarcasm is lost.

Sarcasm can be considered as expressing a *bitter gibe* or *taunt*. Examples include statements such as “Is it time for your medication or mine?” and “I work 40 hours a week to be this poor”.

Sarcasm detection is a binary classification problem. The labeled datasets available to researchers allow Linear Model classification algorithms to be used. The most popular algorithm, among those who chose Machine Learning over Rule-Based approach, is Support Vector Machines (SVM). Logistic

Regression and Naive Bayes are the next most popular classifiers, followed by Decision Tree algorithm. Researchers compare the performance of different classifiers and combine them together to achieve a better sarcasm detection score.

To understand and detect sarcasm 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 the pain of breakup", it is difficult to extract the knowledge needed to detect if there is sarcasm in this statement. In the example, "I love the pain" provides knowledge of the sentiment expressed by the author (in this case positive), and "breakup" describes a contradicting sentiment (that of negative).

Other challenges that exist in understanding sarcastic statements is the reference to multiple events and the need to extract a large amount of facts, commonsense knowledge, anaphora resolution, and logical reasoning.

Sarcasm detection relies on the assumption that a negative situation often appears after the positive situations in a sarcastic document. (document here refers to text, or a tweet)

A sarcastic sentence is formed by a positive verb phrase and a negative verb phrase.

2. Dataset

The dataset used in this work is the news headline dataset, which is found here :
https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection

The dataset is stored in a JSON format, which consists of key-value pairs. There are three keys in the file, the article link, the article headline, and is_sarcastic, which consists as its value, a 0 or a 1, depending on whether a text is non-sarcastic, or sarcastic, respectively. The headlines are obtained from TheOnion, which aims at producing sarcastic (1) versions of the headlines of current affairs, and HuffPost, which consists of real headlines, which are non-sarcastic (0).

The only columns that are used here in the work are "article headline", and "is_sarcastic".

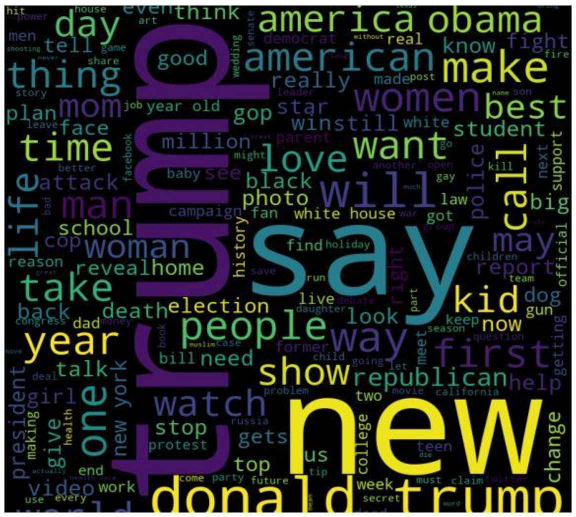

Figure 1 : Word Cloud for all Sarcastic Headlines

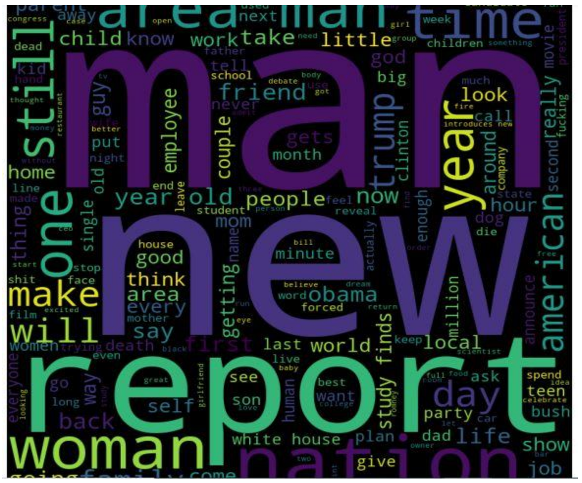

Figure 2 : Word Cloud for all Non - Sarcastic Headline

In the given work, only the headlines are analyzed for the detection of sarcasm, as a whole article itself cannot be sarcastic, given that it is an official news article.

Usually, for sarcasm detection in a text, tweets from Twitter users are extracted from the website itself, and then analysed. However, a major challenge faced when using tweets as a dataset, is the usage of bad language, in terms of grammar, and the occurrences of hashtags, tags, and various other noisy elements. The main advantage of using the news headlines as a dataset for detection of sarcasm is the fact that since news headlines are written by professionals in a formal manner, there are no spelling mistakes and informal usage. This reduces the sparsity and also increases the chance of finding pre-trained embeddings. Furthermore, since the sole purpose of *TheOnion* is to publish sarcastic news, we get high-quality labels with much less noise as compared to

Twitter datasets. Unlike tweets which are replies to other tweets, the news headlines we obtained are self-contained. This would help us in tearing apart the real sarcastic elements.

3. Literature Survey

Past studies in Sarcasm Detection mostly make use of Twitter datasets collected using hashtag based supervision but such datasets are noisy in terms of labels and language. Furthermore, many tweets are replies to other tweets and detecting sarcasm in these requires the availability of contextual tweets.

(Amir et al., 2016) propose to use a CNN to automatically extract relevant features from tweets and augment them with user embeddings to provide more contextual features during sarcasm detection. Twitter dataset used in the study was collected using hashtag based supervision. As per various studies [(Liebrecht et al., 2013; Joshi et al., 2017)], such datasets have noisy labels. Furthermore, people use very informal language on twitter which introduces sparsity in vocabulary and for many words pre-trained embeddings are not available. Lastly, many tweets are replies to other tweets and detecting sarcasm in these requires the availability of contextual tweets. The modeling proposed is quite simplistic. Authors use CNN with one convolutional layer to extract relevant features from text which are then concatenated with (pretrained) user embeddings to produce the final classification score. However, some studies like (Yin et al., 2017) show that RNNs are more suitable for sequential data. Furthermore, authors propose a separate method to learn the user embeddings which means the model is not trainable end to end. Authors do not provide any qualitative analysis from the model to show where the model is performing well and where it is not. Upon analysis, we understand that detecting sarcasm requires understanding of common sense knowledge without which the model might not actually understand what sarcasm is and just pick up some discriminative lexical cues. This direction has not been addressed in previous studies to the best of our knowledge.

In the work by (Shubham Bhasker et al., 2019) a machine learning classifier is used for the classification of news into fake or genuine, and the hybridisation is applied in choosing the features, that is, the content-based and social media based features of the news article, are chosen, and a remarkable result with a very high frequency is obtained.

In our work presented here, we have not taken the neural networks approach. Instead, we have used the simple machine learning classifiers, and finally obtain a very high accuracy model, which gives us the same dataset, but with values predicted, at a high accuracy, reaching about 83%. The original dataset is split into test and train, at each stage of classification, but is slightly modified by replacing the original test target labels, by the ones predicted using the particular classifier. This step is repeated, upto 4 times, since we use four classifiers, and finally, the highest accuracy model is obtained. We call it hybrid, because of the usage of multiple classifiers, on a single dataset, and hence, producing accuracies, as we proceed further ahead, in the flow of algorithms.

4. Network Architecture

The work as proposed here consists of a combination of classifiers, which train the output of the preceding classifier, and pass on their respective output to the next classifier function. The code is designed in a way such that the accuracy of predictions keeps increasing with the flow of classifiers.

The overall architecture of the network is as follows :

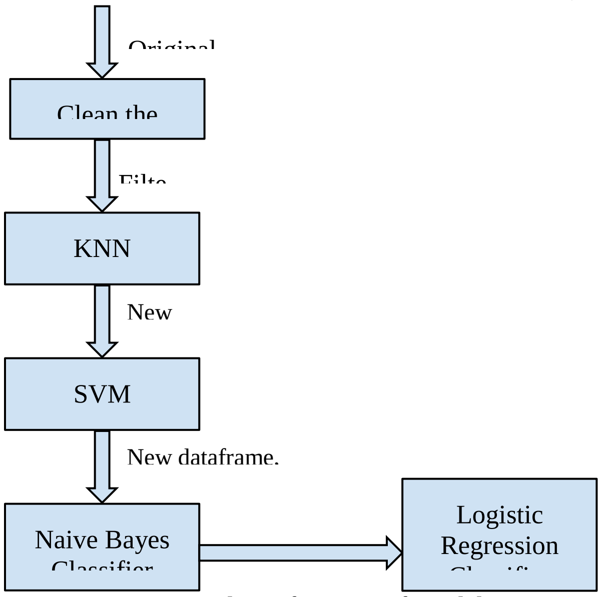

Figure 3 : Flow of Design of Model

The given work has various functions, which makes the entire code easy to read and understand. We first begin with reading the dataset, which shows that each record has three attributes :

- Is_sarcastic : 1 if sarcastic, and 0 if not sarcastic.
- Headline : The headline of the news article
- Article_link : Link to the original news article, useful for collection of supplementary data.

Once the dataset is obtained, we proceed further and start to preprocess the data. Since the dataset consists of news headlines, it does not contain a lot of noise, and is very formally written, and hence does not need an extensive amount of cleaning.

Once preprocessing is done, we move on to start training the data on different classifiers, thereby increasing the accuracy of predictions.

As the diagram suggests above, we start with the K-nearest neighbor classification, split the original dataset into 80:20 ratio, and create a dataframe, such that the column "headlines" consists of those in the test set, and the column "is_sarcastic" consists of all the predicted values of this target label, as obtained from this classifier.

This same procedure of creation of new dataset with the test data and its predictions, replacing the original test data is created for the respective classifier, where the column "headlines" is the same as that of test set and the column "is_sarcastic" consists of the original training labels, and predicted values as obtained as outputs in the previous classifier, and passing this newly created dataset into a new classifier, continues for three classifiers, that is, KNN to SVM, to Naive Bayes, and then finally to Logistic Regression.

Initially, while creating a model like above, the flow of code was different, with the original dataset being passed into Naive Bayes first, followed by KNN classifier, followed by Random Forest Classifier. However, it was realised that when passing the dataset from Naive Bayes to KNN, the accuracy of the model decreased, instead of increasing, and thus, it made very little sense to continue in the same direction. Thus, after much trial and error, the correct flow of classifiers was obtained, and hence finalised for the given work.

We call this a hybrid model for detection of sarcasm, because of the fact that we use various classifiers, to train and test the initial data, and thereby obtain a final dataset, which consists of the most accurate predictions of whether or not the corresponding headline is sarcastic or not, in the form of 1 or 0, respectively.

5. Implementation

As mentioned before, the proposed work uses a variety of machine learning classification algorithms, which train the dataset, and then predict or estimate whether the text is of the sarcastic tone, or not. The list of classifiers as used in the work are, K Nearest Neighbors, Support Vector Machines, Multinomial Naive Bayes, and lastly, Logistic Regression.

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. A supervised machine learning algorithm (as opposed to an unsupervised machine learning algorithm) is one that relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data. The dataset as used in the proposed work, is a labelled dataset, with the target label being "is_sarcastic", and its values being either 0 or 1. Hence, the KNN classifier can be used for the given dataset.

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

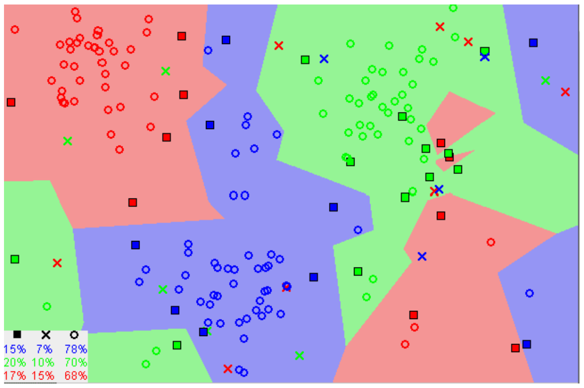

Figure 4 : Image showing similar points exist near each other.

The algorithm for KNN works as follows :

1. Load the data
2. Initialize K to your chosen number of neighbors
3. For each example in the data :
    - Calculate the distance between the query example and the current example from the data.
    - Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distance
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

In our work, the correct value of k is chosen after much trial and error, by running the code using different values of k. It was kept in mind that lower the value of k, less accurate and stable is the prediction, and thus, a value which is quite greater in magnitude must be chosen, as a large value, stabilises the model, and hence gives more accurate results.

The chosen value of k as chosen by us is 7, which gives us an accuracy of about 73%, which is sufficient, for constructing a hybrid model.

Next in the hybrid model, we use the support vector machine classifier. Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. It Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

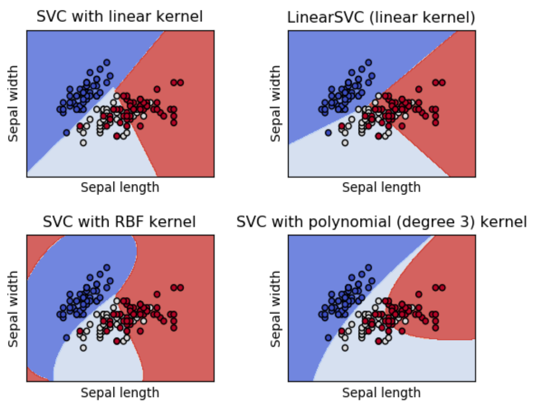

Figure 5 : Different kinds of Kernels and the classifications using SVM

The most common answer is word frequencies. This means that we treat a text as a bag of words, and for every word that appears in that bag we have a feature. The value of that feature will be how frequent that word is in the text. For a more advanced alternative for calculating frequencies, we can also use TF-IDF.
Every text in the dataset is represented as a vector with thousands (or tens of thousands) of dimensions, every one representing the frequency of one of the words of the text. This is what is fed to SVM for training.
In our work, we use the TF-IDF vectorizer, to convert plain text, into vectors. The kernel function as used in our work is a simple linear kernel, as choosing a non-linear kernel will simply overfit the data and hence complicate the solution.
After training the classifier on the training data, we obtain a model, which has an accuracy of about 78%, which is satisfactory for building a hybrid model.
The next classifier as used by us is Naive Bayes Classifier. This classifier is a family of probabilistic algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of a feature.
Bayes theorem calculates probability P(c|x) where c is the class of the possible outcomes and x is the given instance which has to be classified, representing some certain features.

$$\mathbf{P(c|x) = P(x|c) * P(c) / P(x)}$$

Naive Bayes are mostly used in natural language processing (NLP) problems. Naive Bayes predict the tag of a text. They calculate the probability of each tag for a given text and then output the tag with the highest one.

The important part is to find the features from the data to make machine learning algorithms work. In this case, we have text. We need to convert this text into numbers that we can do calculations on. We use word frequencies. That is treating every document as a set of the words it contains. Our features will be the counts of each of these words. This is called feature engineering.
In our work, we convert the words in text into features/vectors, by using the bag of words concept which calculates the frequency of each word in the text.
The usage of Naive Bayes increases the accuracy to 81.3%, which is satisfactory for our hybrid model construction.
The final classifier as used in the work is the Logistic Regression Classifier. **Logistic Regression** is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X. Even though the dataset used in the work does not contain categorical dependent variables, it is binary in nature, and does have the value of either 0 or 1, giving information of whether it is non-sarcastic, or sarcastic in nature. It is the most important (and probably most used) member of a class of models called generalized linear models. Unlike linear regression, logistic regression can directly predict probabilities (values that are restricted to the (0,1) interval); furthermore, those probabilities are well-calibrated when compared to the probabilities predicted by some other classifiers, such as Naive Bayes. Logistic regression preserves the marginal probabilities of the training data. The coefficients of the model also provide some hint of the relative importance of each input variable.

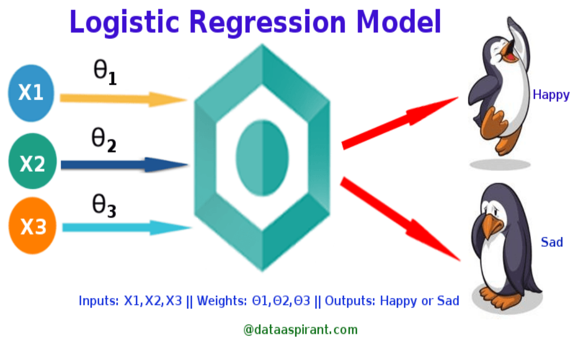
Logistic Regression Model
X1
X2
X3
θ1
θ2
θ3
Happy
Sad
Inputs: X1,X2,X3 || Weights: Θ1,Θ2,Θ3 || Outputs: Happy or Sad
@dataaspirant.com

Figure 6 : Example of Logistic Regression

The underlying algorithm of Maximum Likelihood Estimation (MLE) determines the regression coefficient for the model that accurately predicts the probability of the binary dependent variable. The algorithm stops when the convergence criterion is met or maximum number of iterations are reached. Since the probability of any event lies between 0 and 1 (or 0% to 100%), when we plot the probability of dependent variable by independent factors, it will demonstrate an ‘S’ shape curve.

Logit Transformation is defined as follows-

Logit = Log (p/1-p) = log (probability of event happening/ probability of event not happening) = log (Odds)

Logistic Regression is part of a larger class of algorithms known as Generalized Linear Model (GLM). The fundamental equation of generalized linear model is:

$$\mathbf{g(E(y)) = \alpha + \beta x_1 + \gamma x_2}$$

Here, g() is the link function, E(y) is the expectation of the target variable and $\alpha + \beta x1 + \gamma x2$ is the linear predictor (α,β,γ to be predicted). The role of link function is to ‘link’ the expectation of y to linear predictor.

In our work as presented, we have used logistic regression to classify whether a text is sarcastic or not. The dataset that we are working on here is a combination of the original training set and the test set and the predictions of whether the text is sarcastic or not, as the target labels of the corresponding test set, as obtained from the Naive Bayes Model. It has only the columns that we are in need of, that is, the “headlines” which is the independent variable, and “is_sarcastic” which is the dependent variable. We start by first writing a linear regression equation,

$$\mathbf{g(y) = \beta o + \beta(Headlines)} \text{ — — (a)}$$

g(y) is the link function, and the dependent variable is enclosed in it.

In logistic regression, we are only concerned about the probability of outcome dependent variable (sarcastic or non-sarcastic). As described above, g() is the link function. This function is established using two things: Probability of Sarcastic(p) and Probability of Non-Sarcastic(1-p). p should meet following criteria:

1. It must always be positive (since p >= 0)

2. It must always be less than equals to 1 (since $p <= 1$)

To establish link function, we denote g() with 'p' initially and eventually end up deriving this function. Since probability must always be positive, we'll put the linear equation in exponential form. For any value of slope and dependent variable, the exponent of this equation will never be negative.

p = exp(βo + β(Headlines)) = e^(βo + β(Headlines)) — — — — (b)

To make the probability less than 1, divide p by a number greater than p.
This can simply be done by:

p = exp(βo + β(Headlines)) / exp(βo + β(Headlines)) + 1 = e^(βo + β(Headlines)) / e^(βo + β(Headlines)) + 1 — — — (c)

Using (a), (b) and (c), we can redefine the probability as:

p = e^y/ 1 + e^y — — (d)

where p is the probability of success. *This (d) is the Logit Function*
If p is the probability of success, 1-p will be the probability of failure which can be written as:

q = 1 — p = 1 — (e^y/ 1 + e^y) — — (e)
where q is the probability of failure

On dividing, (d) / (e), we get,

p/(1-p) = e^y

After taking *log* on both side, we get,

log(p/(1-p)) = y

log(p/1-p) is the link function. Logarithmic transformation on the outcome variable allows us to model a non-linear association in a linear way.
After substituting value of y, we'll get:

log(p/(1-p)) = βo + β(Headlines)

This is the equation used in Logistic Regression.
The accuracy as obtained when we train this classifier on the dataset, is about 83%.
When performing all of the classifications on the original dataset, as read from the JSON file, we obtain certain accuracies, which are plotted in the bar chart, as given below. While comparing the two bar plots given below, we can come to the conclusion that hybridizing the model, by combining various classifiers, and then modifying the dataset, as we do, we increase the stability, as well as the accuracy of the output, as obtained finally.

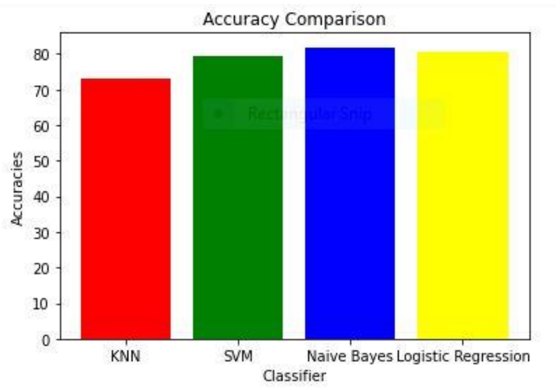

Figure 7 : Accuracies when the classifiers are trained on the same dataset.

The tabular comparison of the accuracies of classifiers, when they are trained on the original dataset, without actually modifying it within the classifier module itself, as are follows :

	KNN Classifier	SVM Classifier	Naive-Bayes Classifier	Logistic Regression
Accuracy (in %)	73	79	82	80

Table 1 : Comparative Accuracies of individual classifiers, on original dataset

Tabular comparison of the accuracies of the hybrid model, as constructed in the produced work, are as given below :

	KNN Classifier	SVM Classifier	Naive-Bayes Classifier	Logistic Regression
Accuracy (in %)	73	78	81	83

Table 2 : Comparative Accuracies of the Model

The various classifiers' accuracies, in the hybrid model, can be compared as follows :

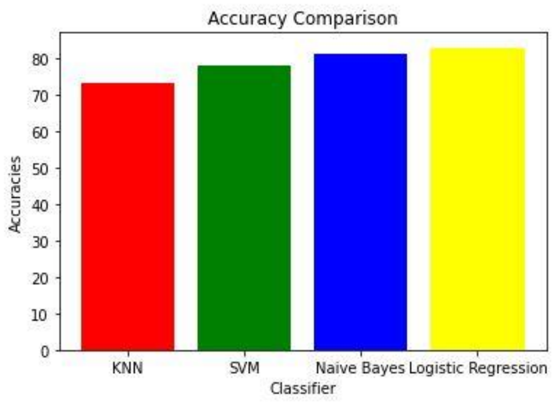

Figure 8 : Comparison of Accuracies for various classifiers

As aforementioned, the two tables as given above, compare the accuracies of predictions as found by the four classifiers, however, the former, trains the individual classifiers on the original dataset, whereas the latter trains the classifiers, individually, on modified dataset, with the testing labels, being replaced by the predicted labels by that particular classifier. The accuracy comparison tables above basically gives the reader an idea, about how the proposed work, increases the accuracy of predictions, with the flow of the program. If we test the classifiers individually, in the same order, as in the hybrid model constructed, it is observed that the accuracy as obtained for each, either remains the same, or is decreased. A major observation here is that in the normal flow of classification using different classifiers on the original dataset, naive bayes classifier gives the maximum accuracy, which decreases in the case of the last classifier, logistic regression. In the hybrid model, the classifier modules are arranged in a way such that, naive bayes gives an accuracy of 81%, which is high enough, and later, when logistic regression is used for classification on the modified dataset, as obtained from the naive bayes model, the accuracy increases to 83%, unlike in the case of the individual classifiers training, on the original dataset obtained.
For the different classifiers as used in the work, we obtain a classification report for each, which gives us a clear idea, of the different measures of correctness of the different models.

Target Label	Precision	Recall	F1 - Score	Accuracy
0	0.71	0.89	0.79	
1	0.78	0.52	0.62	0.73

Table 3 : Classification Report of the KNN Model

Target Label	Precision	Recall	F1 - Score	Accuracy
0	0.79	0.85	0.82	0.78
1	0.76	0.68	0.72	

Table 4 : Classification Report of the SVM Model

Target Label	Precision	Recall	F1 - Score	Accuracy
0	0.83	0.86	0.85	0.81
1	0.78	0.74	0.76	

Table 5 : Classification Report for the Naive-Bayes Model

Target Label	Precision	Recall	F1 - Score	Accuracy
0	0.83	0.89	0.86	0.83
1	0.82	0.73	0.77	

Table 5 : Classification Report for the Logistic Regression Model

7. Conclusion

The flow of the work starts with getting the dataset in the JSON format, which has three columns, the headline, the original article link, and a target column, called “is_sarcastic” which defines whether a headline is sarcastic or not. The proposed work requires only two of the columns, so the column which lists the original article links is discarded. This is followed by preprocessing of the column which contains the headlines alone, by lemmatizing the text, and then removing the stop words. The target column, is_sarcastic is left untouched.

Once preprocessing is done, the original dataset, with filtered headlines, is passed into the first classifier, that is KNN in the model constructed, and is split into training and test dataset. The classifier is trained on the training set, and then performs the classification on the test set of headlines, and predicts values for their corresponding target labels (“is_sarcastic”) in the form of either a 0, or 1. In the hybrid model as constructed here, the KNN classifier module then returns a dataset, which consists of the original training set, along with the target values, as well as the headlines in the test set, however, the target labels for these test set headlines are the ones as predicted by the KNN classifier. Thus, a new dataset is obtained which consists of a training set, with the same original target labels, combined with the test set headlines, with the newly predicted labels for the same. This same modification is done through all classifiers, that is KNN is followed by SVM, then Naive Bayes, and finally, Logistic Regression.

In conclusion, a hybrid model is obtained, in which the hybridisation is done, not on the classifiers, but on the dataset itself, where we obtain datasets after modifications performed as mentioned above. This hybridisation finally outputs a dataset, which has a highly accurate result, consisting of the news headlines column, and the column “is_sarcastic”, which defines whether a text is sarcastic or not. This dataset, on an overall basis, consists of the predictions made by all the classifiers that it passes through, and thus, gives an output which contains predictions of the various classifiers as the target labels.

Future Works

In the proposed work here, we have used various machine learning classifiers like KNN, SVM, Naive Bayes and Logistic Regression.

Even though the flow of events produces quite a high accuracy, there are other unexplored areas which will produce an even higher accuracy, of upto 95%.

Some of the works, which have been done already, are using artificial neural networks, which are either convolutional neural networks, or recurrent neural networks, or a combination of both.

Given the time crunch, we are left with several unexplored directions that we would like to work on in future. Some of the important directions are as follows:

• We can do ablation study on our proposed architecture to analyze the contribution of each module.

• The approach proposed in this work could be considered as a pre-computation step and the learned parameters could be tuned further on any other dataset, small or large in number. Our intuition behind this direction is that this pre-computation step would allow us to capture the general cues for sarcasm which would be hard to learn on a given dataset alone, if it is small in size. This type of transfer learning is shown to be effective when limited data is available.

• Lastly, we observe that detection of sarcasm depends a lot on common knowledge (current events and common sense). Thus, we plan to integrate this knowledge in our work, along with the introduction of a simple neural network, so that our model is able to detect sarcasm based on which sentences deviate from common knowledge. Recently, (Young et al., 2017) integrated such knowledge in dialogue systems and the ideas mentioned could be adapted in our setting as well.

Acknowledgements

We would like to thank PES University for giving us an opportunity to work on a project like this, which gave us so much knowledge outside of our domains, and we learnt so much while working on the same.

We would like to thank Mrs. Shylaja S.Sharath, Chairman, Department of CSE, PES University, for making it possible for us to write a paper, and propose a work as such,

The proposed work was inspired by the teachings of our professor, Mrs. Ashwini M Joshi. She has guided us through the whole procedure on how to build a hybrid model for text classification, and has been a great mentor throughout.

We would also like to thank Rishabh Mishra, whose work, as available in his repository on GitHub has helped us to a great extent. The way he has explained the procedure of cleaning the text, and building a hybrid model on it, has greatly helped our work, and we are grateful to him for putting his work on a site like GitHub to help others.

9. Contributions and References

Both team members who are a part of this project have contributed equally to the proposed work.

The references as used by us in building the work are as follows :

[A] Silvio Amir, Byron C Wallace, Hao Lyu, and Paula Carvalho Mario J Silva. 2016. Modelling context ´ with user embeddings for sarcasm detection in social media. arXiv preprint arXiv:1607.00976 .

[B] CC Liebrecht, FA Kunneman, and APJ van Den Bosch. 2013. The perfect solution for detecting sarcasm in tweets# not .

[C] Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. 2017. Automatic sarcasm detection: A survey. ACM Computing Surveys (CSUR) 50(5):73.

[D] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of cnn and rnn ¨ for natural language processing. arXiv preprint arXiv:1702.01923 .

[E] Misra, Rishabh & Arora, Prahal. (2018). Sarcasm Detection using Hybrid Neural Network. 10.13140/RG.2.2.32427.39204.

[F] Competitive generative models with structure learning for NLP classification tasks Kristina Toutanova Microsoft Research Redmond, WA

[G] Francis Chantree, Bashar Nuseibeh, Anne De Roeck, and Alistair Willis. 2006. Identifying nocuous ambiguities in natural language requirements. In Proceedings of 14th IEEE International Requirements Engineering conference (RE'06), Minneapolis/St Paul, Minnesota, USA, September.

[H] Bauskar, Shubham & Badole, Vijay & Jain, Prajal & Chawla, Meenu. (2019). Natural Language Processing based Hybrid Model for Detecting Fake News Using Content-Based Features and Social Features. International Journal of Information Engineering and Electronic Business. 11. 1-10. 10.5815/ijieeb.2019.04.01.

[I] Shubham Bauskar, Vijay Badole, Prajal Jain, Meenu Chawla, " Natural Language Processing based Hybrid Model for Detecting Fake News Using Content-Based Features and Social Features", International Journal of Information Engineering and Electronic Business(IJIEEB), Vol.11, No.4, pp. 1-10, 2019. DOI: 10.5815/ijieeb.2019.04.01

[J] Santiago Castro, Devamanyu Hazarika , Verónica Pérez-Rosas, Roger Zimmermann, Rada Mihalcea , Soujanya ,Computer Science & Engineering, University of Michigan, USA School of Computing, National University of Singapore, Information Systems Technology and Design, SUTD, Singapore

[K] E. Riloff, A. Qadir, P. Surve, L. De Silva, N. Gilbert, and R. Huang, "Sarcasm as contrast between a positive Sentiment and negative situation", in Proc. Con Empirical Methods Natural Lang. Process, Oct.2013, pp.704_714.

[L] M. Bouazizi, T. Ohtsuki, "Pattern-Based Approach for Sarcasm Detection on Twitter" VOLUME 4, 10.1109/ACCESS.2016.2594194.

[M] S.K. Bharti B. Vachha, R.K. Pradhan, K.S. Babu, S.K. Jena "Sarcastic sentiment detection in tweets Streamed in real time: a big data approach", Elsevier 12 July 2016.

[N] Evgeny Byvatov, Uli Fechner, Jens Sadowski, and Gisbert Schneider. 2003. Comparison of support vector machine and artificial neural network systems for drug/nondrug classification. Journal of chemical information and computer sciences, 43(6):1882–1889

[O] Gavin Abercrombie and Dirk Hovy. 2016. Putting sarcasm detection into context: The effects of class imbalance and manual labelling on supervised machine classification of twitter conversations. In Proceedings of the ACL 2016 Student Research Workshop, pages 107–113.

[P] Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, Gautam Naik, Erik Cambria, and Rada Mihalcea. 2018. Meld: A multimodal multi-party dataset for emotion recognition in conversations. arXiv preprint arXiv:1810.02508.

[Q] Amir Zadeh, Minghai Chen, Soujanya Poria, Erik Cambria, and Louis-Philippe Morency. 2017. Tensor fusion network for multimodal sentiment analysis. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 1103–1114.

[R] Ashwin Rajadesingan, Reza Zafarani, and Huan Liu. 2015. Sarcasm detection on twitter: A behavioral modeling approach. In Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, pages 97–106. ACM.