Advance Math Project

Sarcasm Detection

Group members

Shaurya Negi

Reshma Madhavankutty

Rishab Mahajan

Krish Shah

Kunal Dubey

INDEX

INTRODUCTION

Sarcasm plays an important role in daily conversations by allowing individuals to express their intent to mock or display contempt. It is achieved by using irony that reflects a negative connotation. For example, in the utterance: Maybe it's a good thing we came here. It's like a lesson in what not to do, the sarcasm is explicit as the speaker expresses learning of a lesson in a positive light when in reality, she means it in a negative way However, there are also scenarios where sarcasm lacks explicit linguistic markers, thus requiring additional cues that can reveal the speaker's intentions. For instance, sarcasm can be expressed using a combination of verbal and non-verbal cues, such as a change of tone, overemphasis in a word, a drawn-out syllable, or a straight looking face. Moreover, sarcasm detection involves finding linguistic or contextual incongruity, which in turn requires further information, either from multiple modalities.

Sarcasm detection is a very narrow research field in NLP, a specific case of sentiment analysis where instead of detecting a sentiment in the whole spectrum, the focus is on sarcasm. Therefore, the task of this field is to detect if a given text is sarcastic or not.

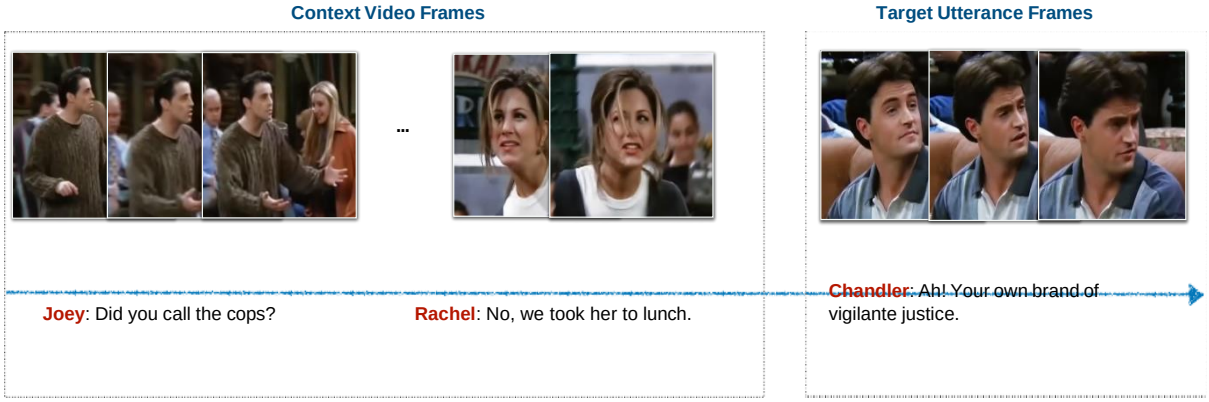


Fig (1)

In the following report, we will discuss the implementation of machine learning models and deep learning models. Also, why we chose deep learning model over machine learning model.

DATASET DESCRIPTION

Previous work in sarcasm detection have primarily relied on Twitter datasets gathered by hashtag supervision, however such datasets are messy in terms of labeling and language. Moreover, many headlines are answers to other headlines, and detecting sarcasm in these necessitates the presence of relevant headlines.

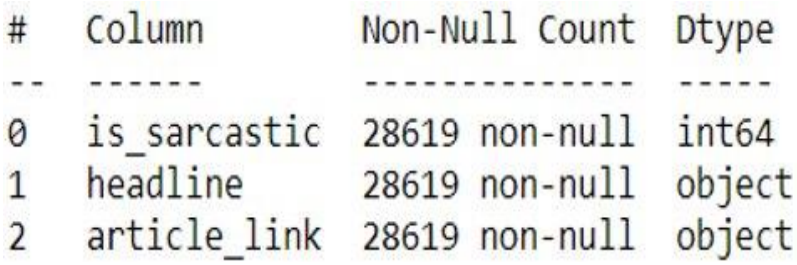
This News Headlines dataset for Sarcasm Detection was obtained from two news websites to address the limits of noise in datasets. The Onion's goal is to create satirical renditions of current events, so we gathered all of the headlines from the News in Brief and News in Photos sections (which are sarcastic). HuffPost provides us with real (non-sarcastic) news headlines.

In comparison to existing Twitter datasets, this new dataset has the following advantages:

There are no spelling mistakes or colloquial use in news headlines because they are written by experts in a formal manner. This minimizes the sparsity of the data while also increasing the likelihood of identifying pre-trained embeddings.

Furthermore, because The Onion's only aim is to provide satirical news, we acquire high-quality labels with significantly less noise than Twitter datasets.

The news headlines we acquired are self-contained, unlike tweets, which are replies to other tweets. This would aid us in separating the genuine sarcastic aspects.



#	Column	Non-Null Count	Dtype
0	is_sarcastic	28619 non-null	int64
1	headline	28619 non-null	object
2	article_link	28619 non-null	object

Fig (2)

EXPLORATORY DATA ANALYSIS

As a basic exploration, we visualize the types of words that occur frequently in each category through Bar charts, world clouds.

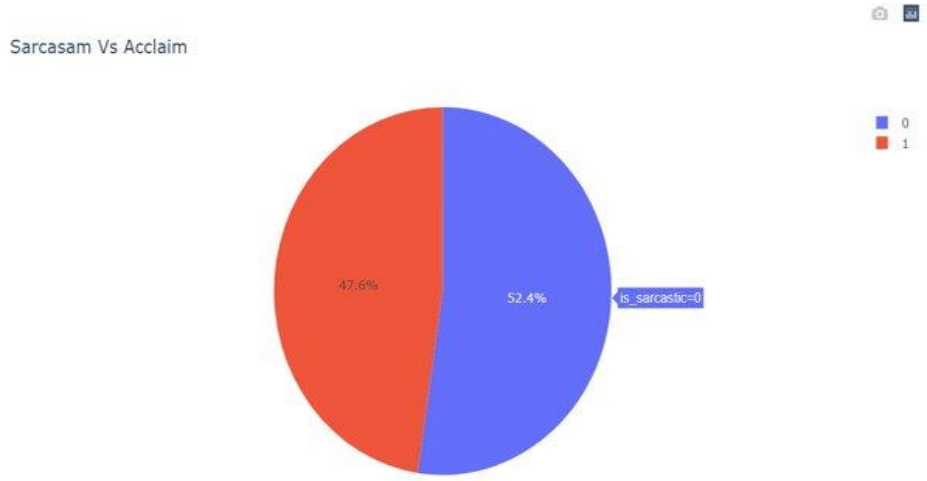


Fig (3)

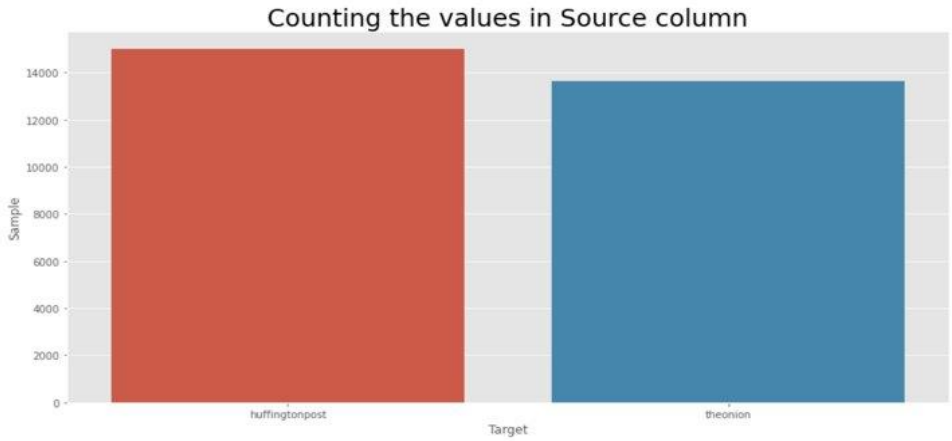


Fig (4)

Frequency of words for whole Dataset

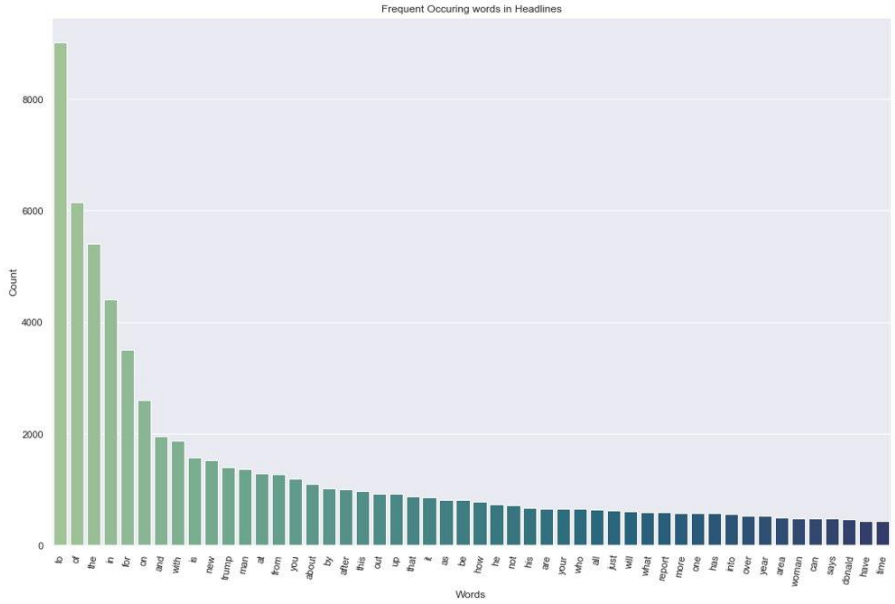


Fig (5)

The above graph shows the common words are to, of, the, on which are the most common words. The common words also show the frequency are the adjectives only.

Frequency Of Words For Sarcastic Headlines

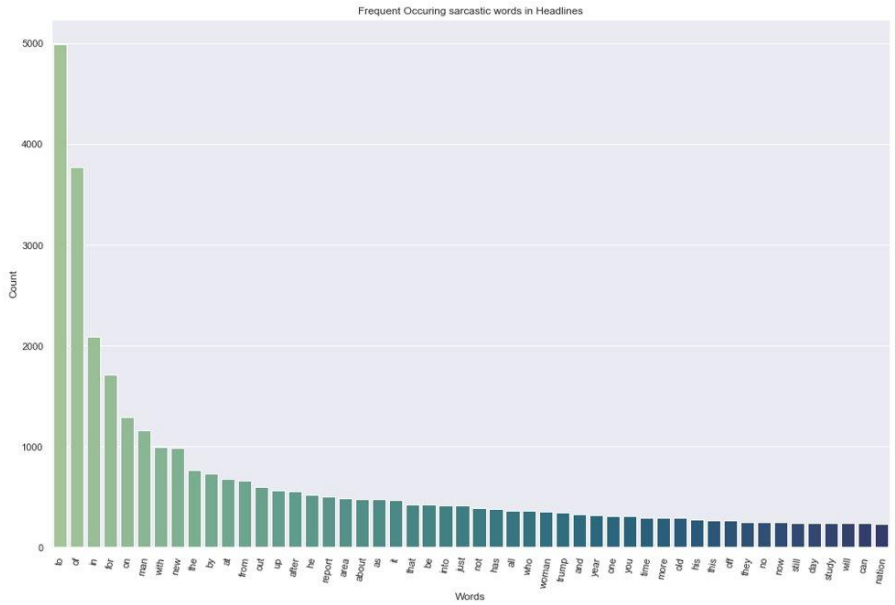


Fig (6)

The Above Graph Shows The Frequency Is Almost Half The Whole Dataset. The Common Adjectives Are To, Of, In And For, Which Is Quite Similar To That The Frequency Of Non-Sarcastic Words.

Frequency of Words for Non-Sarcastic Headlines

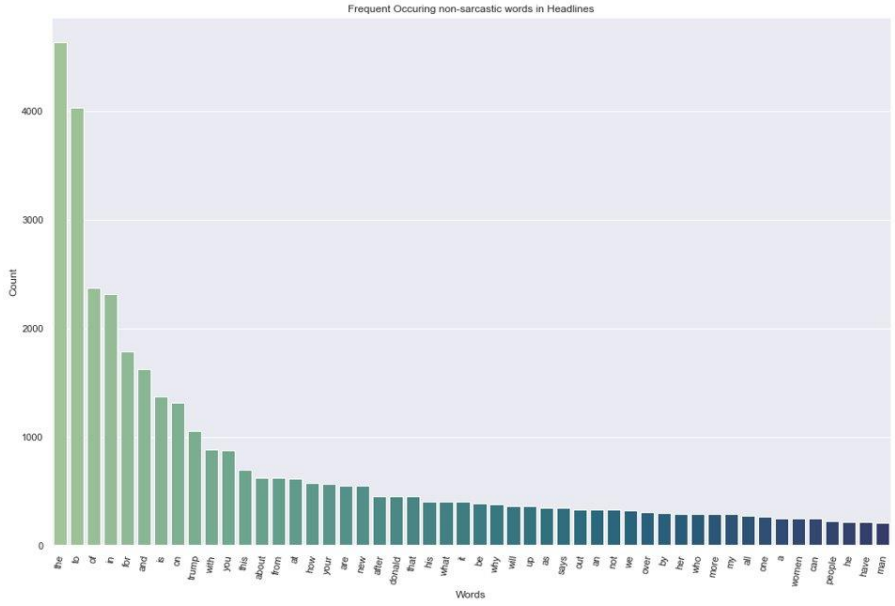


Fig (7)

The trend of the common non sarcastic words is high compared to sarcastic, which is quite expected.

Word Cloud

A word cloud (otherwise called text cloud or label cloud) works in a straightforward way the more a particular word shows up in a wellspring of printed information (like a discourse, blog entry, or data set), the greater and bolder it shows up in the word cloud.

A word cloud is an assortment, or group, of words portrayed in various sizes. The greater and bolder the word shows up, the more regularly it's referenced inside a given text and the more significant it is. Otherwise, called label mists or text mists, these are ideal ways of taking out the most appropriate pieces of printed information, from blog entries to data sets. They can likewise help business clients investigate two distinct bits of text to track down the phrasing similitudes between the two.

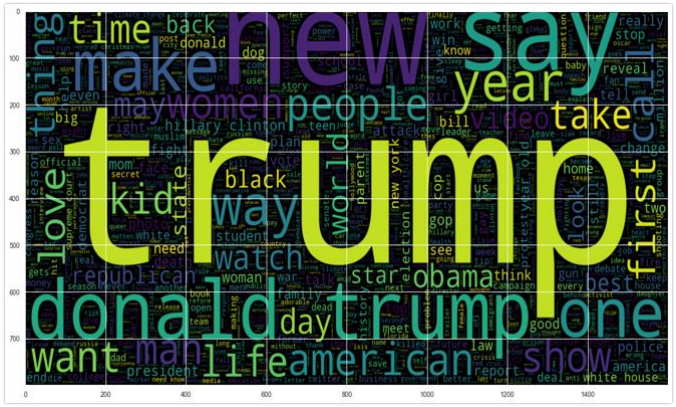


Fig (8)

Fig (9)

Length Of Sentences

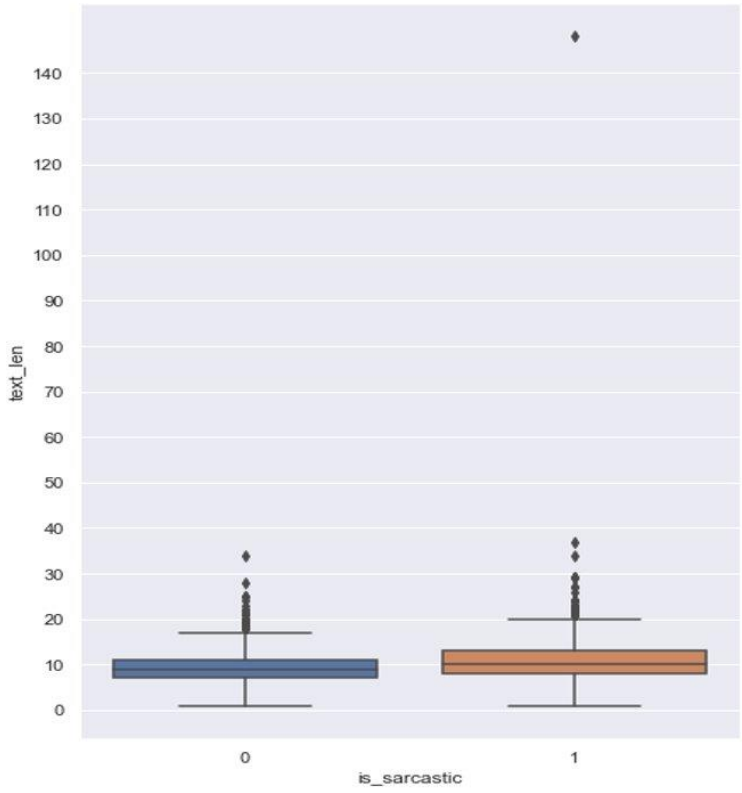


Fig (10)

The length of sentences are same and they are identical as well, but in some rare cases it is extreme as well, which act as an outlier for the model.

MACHINE LEARNING MODELLING

The top 10 features in decreasing order of importance for sarcasm detection are the following:

1. Percentage of emoticons in the headlines.
2. Percentage of adjectives in the headlines.
3. Percentage of past words with sentiment score.
4. Number of polysyllables per word in the headlines.
5. Lexical density of the headlines.
6. Percentage of past words with sentiment score 2.
7. Percentage of past words with sentiment score -3.
8. Number of past sarcastic headlines posted.
9. Percentage of positive to negative sentiment transitions made by the user.
10. Percentage of capitalized hashtags in the tweet.

Pipeline

A machine learning pipeline is utilized to assist with computerizing AI work processes. They work by empowering a grouping of information to be changed and related together in a model that can be tried and assessed to accomplish a result, regardless of whether positive or negative.

AI (ML) pipelines comprise of a few stages to prepare a model. AI pipelines are iterative as each progression is rehashed to ceaselessly work on the precision of the model and accomplish an effective calculation. To construct better AI models, and get the most worth from them, available, adaptable and sturdy stockpiling arrangements are basic, preparing for on-premises object stockpiling.

The primary target of having a legitimate pipeline for any ML model is to practice command over it. An efficient pipeline makes the execution more adaptable. It resembles having a detonated perspective on a PC where you can pick the defective pieces and supplant it-for our situation, supplanting a lump of code.

The term ML model alludes to the model that is made by the preparation cycle.

The learning calculation finds designs in the preparation information that map the information ascribes to the objective (the response to be anticipated), and it yields a ML model that catches these examples.

A model can have numerous conditions and to store every one of the parts to ensure all elements accessible both disconnected and online for organization, all the data is put away in a focal vault.

A pipeline comprises of a succession of parts which are an assemblage of calculations. Information is sent through these parts and is controlled with the assistance of calculation.

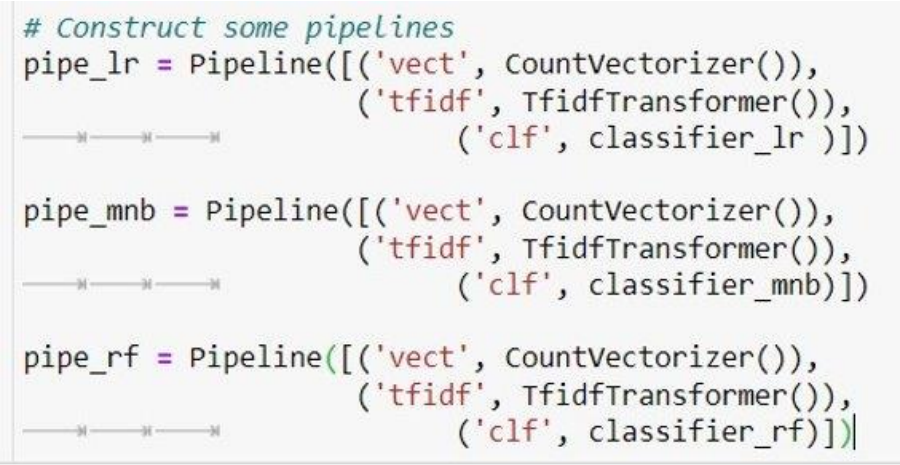


Fig (11)

The machine learning models were trained in logistic regression, multinomial naïve bayes and random forest.

1. Logistic Regression

   Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. It is one of the simplest ML algorithms that can be used for various classification problems.

2. Multinomial Naïve Bayes
   Multinomial Naïve Bayes is based on Bayes theorem. It predicts the likelihood of each category and outputs the one with the highest probability.

3. Random forest
   Random Forest is combination of many decision trees. Output decided by Random Forest is the class which is selected by most trees. It generally outperforms decision trees.

ROC curve

AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. By analogy, the Higher the AUC, the better the model is at distinguishing between patients with the disease and no disease.

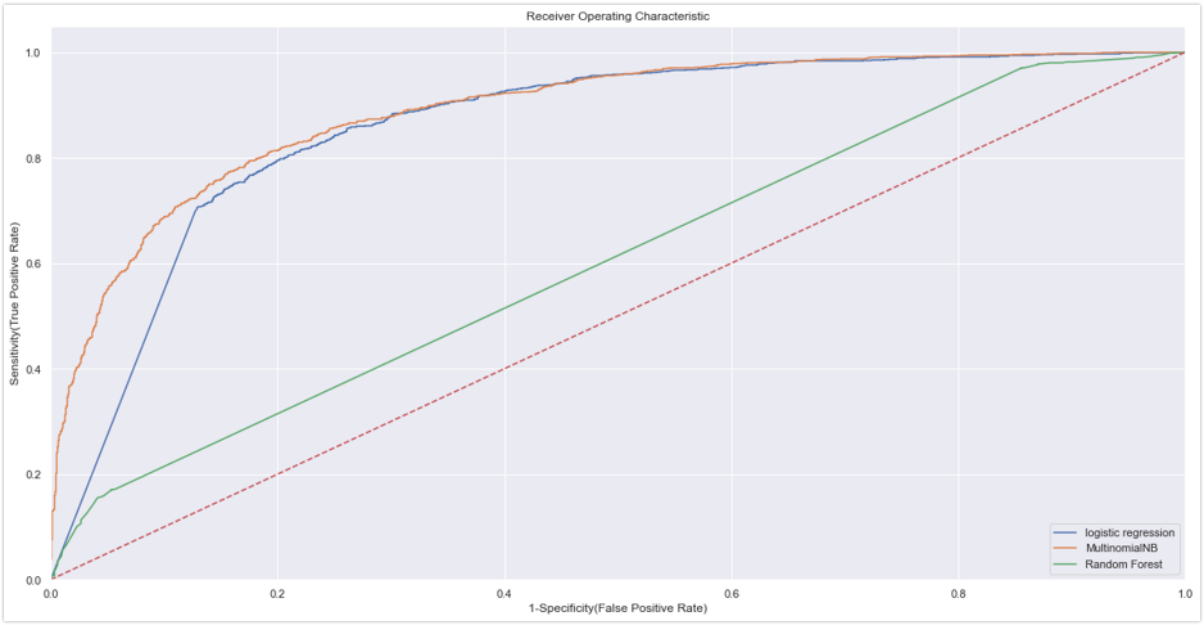


Fig (12)

Here it is evident that Multinomial Naïve Bayes model has highest ROC-AUC score, which depicts that MBN has highest prediction percentage and is effective compared to RF and LR.

Confusion Matrices

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. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

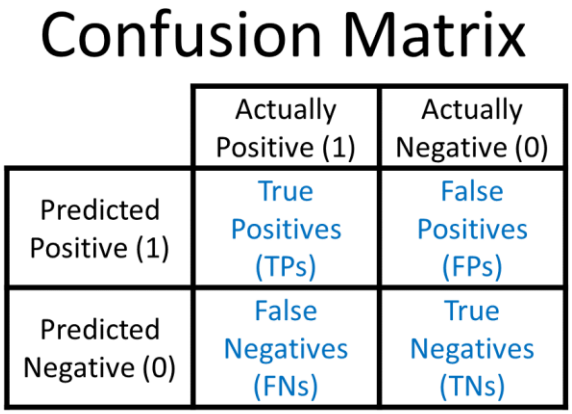


Fig (13)

1. Logistic Regression confusion matrix:

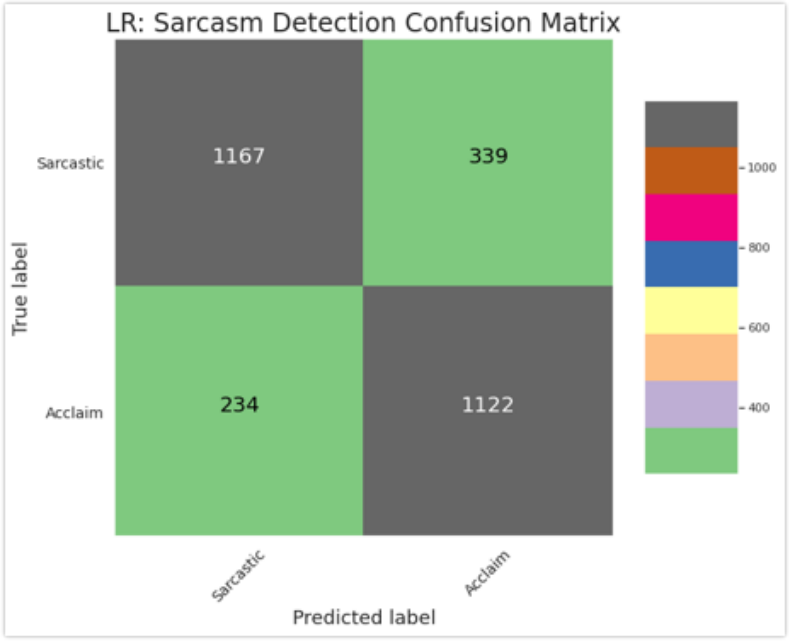


Fig (14)

2. Random Forest confusion matrix

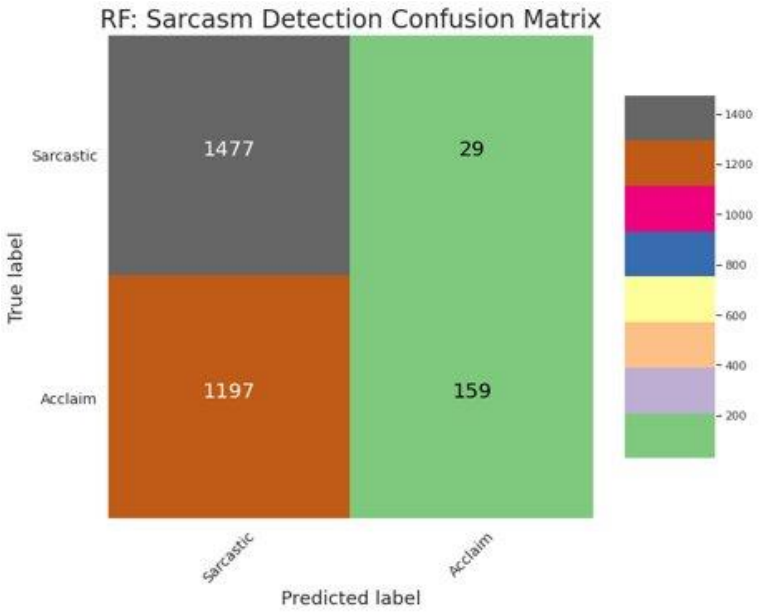


Fig (15)

3. Multinominal naïve bayes Confusion Matrix

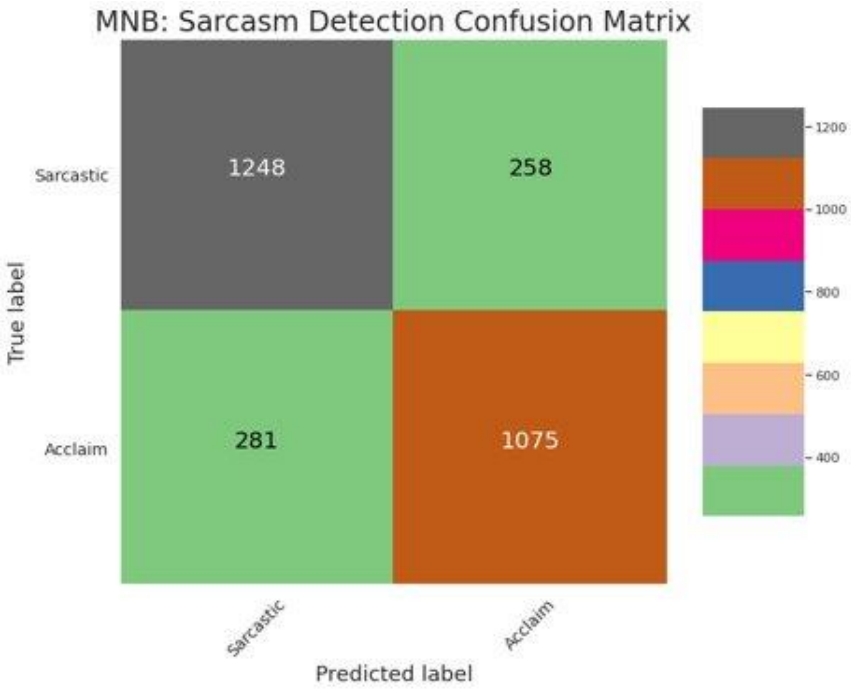


Fig (16)

Observation:

From the above confusion matrices, it is evident that MNB has the best prediction and has least deviation from the detection.

Results for Different ML Models

	Logistic Regression	MultinomialNB	Random Forest
Train	1.00000	0.994798	0.566759
Test	0.79979	0.811670	0.563592

Fig (17)

After training the data using all the three models, we can conclude that Multinomial NB performs comparatively better than other two. Whereas, Random Forest performs very poorly for Sarcasm Detection task. Now, we further try to improve the modelling by using Deep Learning.

DEEP LEARNING MODELS

Sarcasm detection can be one of main implementation using a combination of convolutional neural networks (CNNs). The detection of sarcasm is important as it can help in sentimental analysis which can be help us to understand the context and polarity of the expression.

The main challenge is 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).

The other challenges are frame of reference, common knowledge and logical reasoning of expression it refers to.

We built our Deep Learning Models in separate notebook and environment to avoid any dependencies.

After Pre-processing the dataset, we split the dataset in Training and Validation sets.

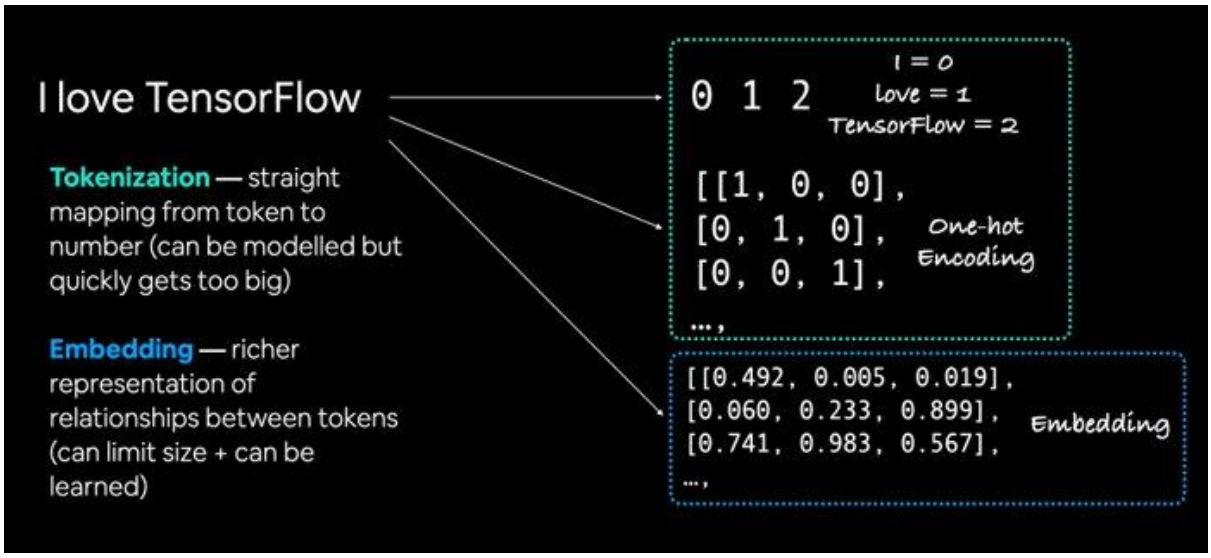


Fig (18)

```
Original text:
the conservative reform movement's raging contradiction

Embedded version:

<tf.Tensor: shape=(1, 30, 128), dtype=float32, numpy=
array([[[-0.04364428,  0.02437404, -0.03696011, ..., -0.04763393,
          0.02931459,  0.0068561 ],
        [-0.00057185, -0.02672013, -0.02819778, ...,  0.019741  ,
         -0.03810944, -0.01872007],
        [ 0.02886024, -0.04978222,  0.00995754, ..., -0.01021207,
          0.04130098,  0.02583535],
        ...,
        [ 0.01645621, -0.00589932, -0.01471175, ..., -0.02511839,
          0.00912381, -0.00024097],
        [ 0.01645621, -0.00589932, -0.01471175, ..., -0.02511839,
          0.00912381, -0.00024097],
        [ 0.01645621, -0.00589932, -0.01471175, ..., -0.02511839,
          0.00912381, -0.00024097]]], dtype=float32)>
```

Each token in the sentence gets turned into a length 128 feature vector.

Fig (19)

Baseline Model

The baseline model can be made using either Convolutional neural network (CNN) and Recurring neural network. We couldn't use RNN as it mainly deals with visual imagery. So, we decided to use CNN, as deal with the sequential data and is efficient in sequence prediction.

But with CNN, key challenges are storing the memory for long term. Therefore we upgraded to Long Short Term Memory (LSTM).

We can think of LSTM as an RNN with some memory pool that has two key vectors:

(1) Short-term state: keeps the output at the current time step.

(2) Long-term state: stores, reads, and rejects items meant for the long-term while passing through the network.

LSTM is the extension of RNN. But it deals with the two major problems, vanishing and exploding gradients. The vanishing gradients deals with slow convergence and exploding gradients which causes too much change in weights. The LSTM keeps, store, reads and rejects in long term.

LSTM introduces a memory cell (or cell for short) that has the same shape as the hidden state (some literatures consider the memory cell as a special type of the hidden state), engineered to record additional information. To control the memory cell we need a number

of gates. One gate is needed to read out the entries from the cell. We will refer to this as the output gate. A second gate is needed to decide when to read data into the cell. We refer to this as the input gate. Last, we need a mechanism to reset the content of the cell, governed by a forget gate.

In Bidirectional LSTM, instead of training a single model, we introduce two. The first model learns the sequence of the input provided, and the second model learns the reverse of that sequence. Since we do have two models trained, we need to build a mechanism to combine both.

Why did we decide to use Bi-LSTM instead of LSTM?

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems.In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence.

```
# Set random seed and create embedding layer (new embedding layer for each model)
tf.random.set_seed(42)
from tensorflow.keras import layers # import layers from Keras
baseline_model_embedding = layers.Embedding(input_dim=max_vocab_length, # set input shape
                                    output_dim=128, # set size of embedding vector
                                    embeddings_initializer="uniform", # default, intialize randomly
                                    input_length=max_length, # how long is each input
                                    name="embedding_2") # name of the embedding layer


# Create Bidirectional LSTM model
inputs = layers.Input(shape=(1,), dtype="string") # input layer
x = text_vectorizer(inputs) # convert input to numerical representation
x = baseline_model_embedding(x) # pass through embedding layer
print(x.shape)
x = layers.Bidirectional(layers.LSTM(128, return_sequences=True))(x) # return vector for each word in the Tweet (you can sta
x = layers.Dropout(0.2)(x) # adding a dropout layer to prevent overfitting
x = layers.BatchNormalization()(x) # normalize the output of the RNN
x = layers.Bidirectional(layers.LSTM(64))(x) # return vector for whole sequence
x = layers.Dropout(0.2)(x) # adding a dropout layer to prevent overfitting
x = layers.BatchNormalization()(x) # normalize the output of the RNN
print(x.shape)
x = layers.Dense(64, activation="relu")(x) # optional dense layer on top of output of LSTM cell
outputs = layers.Dense(1, activation="sigmoid")(x) # output layer
baseline_model = tf.keras.Model(inputs, outputs, name="baseline_model_LSTM") # create model
```

Fig (20)

```
# Calculate LSTM model results
baseline_model_results = calculate_results(y_true=val_labels,
                                           y_pred=baseline_model_preds)
baseline_model_results
```

```
{'accuracy': 86.44304682040531,
 'f1': 0.8644692006156373,
 'precision': 0.866218465268858,
 'recall': 0.8644304682040531}
```

Fig (21)

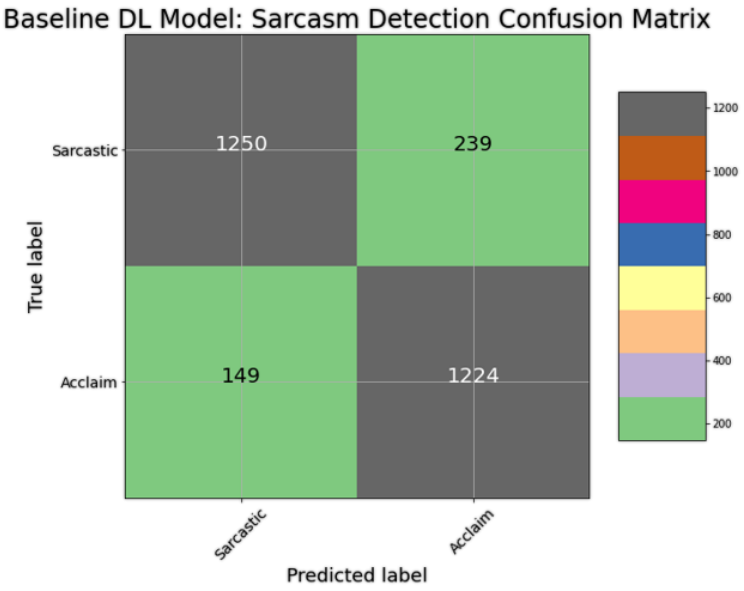
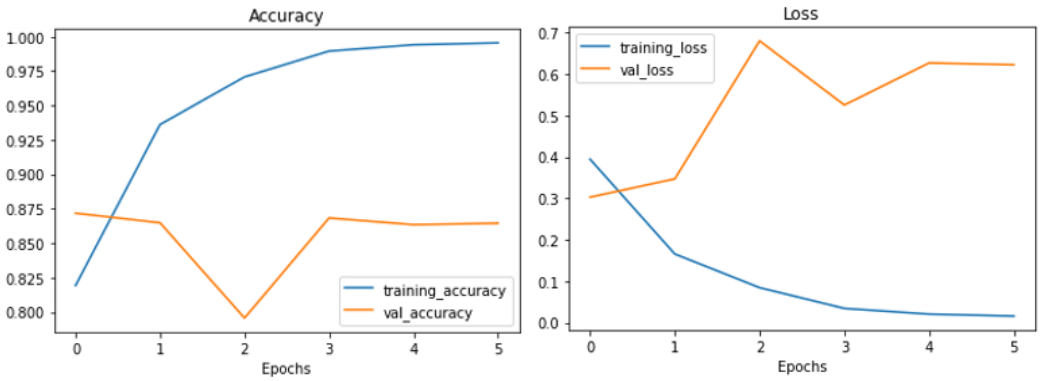


Fig (22)



Fig(23)

Observation:

The training loss is decreasing which is good but the validation loss is stagnant. Also the accuracy is achieved 99% which makes this case of overfitting.

OUR MODEL: MODELLING USING UNIVERSAL SENTENCE ENCODER

The main difference between the embedding layer we created, and the Universal Sentence Encoder is that rather than create a word-level embedding, the Universal Sentence Encoder, as you might've guessed, creates a whole sentence-level embedding.

```
# Example of pretrained embedding with universal sentence encoder - https://tfhub.dev/google/universal-sentence-encoder/4
import tensorflow_hub as hub
embed = hub.load("https://tfhub.dev/google/universal-sentence-encoder/4") # Load Universal Sentence Encoder
embed_samples = embed([sample_sentence,
                       "When you call the universal sentence encoder on a sentence, it turns it into numbers."])

print(embed_samples[0][:50])
```

```
tf.Tensor(
[ 0.02285576  0.05599008 -0.02142934 -0.03681325 -0.01647731  0.01036584
  0.01865435  0.01320562  0.06016935 -0.0036779  -0.03153399 -0.00575823
  0.03084559  0.04710538  0.05997398 -0.05374786 -0.04809381 -0.02493408
  0.00567767 -0.00865088 -0.08051635 -0.05007069  0.05378757  0.01312216
 -0.02426482  0.01372254  0.05393068 -0.01322597  0.06902386 -0.04406415
  0.02934264 -0.05961558 -0.01963601  0.00949474  0.04088591  0.05860829
 -0.01390968 -0.02457503 -0.00445435 -0.04764604  0.08532364 -0.00186264
 -0.00492015  0.02860365 -0.01262604  0.00661173 -0.04614464  0.06792253
  0.02672658 -0.04150194], shape=(50,), dtype=float32)
```

Fig (24)

RESULTS:

	accuracy	precision	recall	f1
baseline	86.443047	0.866218	0.864430	0.864469
tf_hub_sentence_encoder	83.018868	0.830539	0.830189	0.830247

Fig (25)

```
# Calculate model 6 performance metrics
model_USE_results = calculate_results(val_labels, model_USE_preds)
model_USE_results # view the results
```

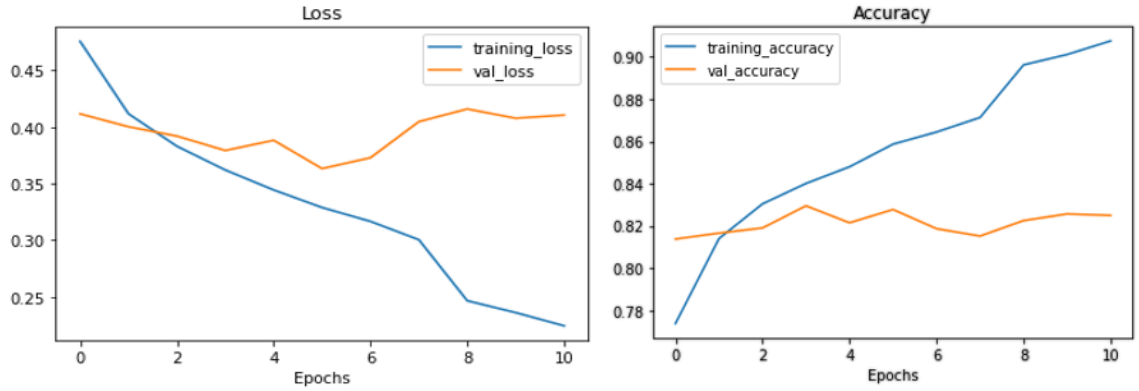
```
{'accuracy': 83.01886792452831,
 'f1': 0.8302465257080635,
 'precision': 0.8305387042980924,
 'recall': 0.8301886792452831}
```

```
# Compare TF Hub model to baseline
compare_baseline_to_new_results(baseline_model_results, model_USE_results)
```

```
Baseline accuracy: 86.44, New accuracy: 83.02, Difference: -3.42
Baseline precision: 0.87, New precision: 0.83, Difference: -0.04
Baseline recall: 0.86, New recall: 0.83, Difference: -0.03
Baseline f1: 0.86, New f1: 0.83, Difference: -0.03
```

Fig (26)

After running using both the models,



Fig(27)

The Validation loss is decreasing and accuracy is also which is evident that our model is progressed and is giving the realistic results.

In this experimental study, three machine learning algorithms were used. These algorithms are LR, RF and MNB. All these algorithms were applied to the UCI Mushroom Classification Dataset. Data was divided into two portions, training data, and testing data, both these portions consisting of 90% and 10% data respectively

We then introduced deep learning because exponential rise of data, the ability to process large numbers of features makes deep learning very powerful when dealing with unstructured data.Deep learning algorithms try to learn high-level features from data.

We used embedding of words which helped in making word cloud. Our baseline model initially relied on CNN which upgraded to LSTM and finally we implemented bidirectional LSTM, which made our model further accurate and efficient.

Even though there was slight overfitting, we used batch normalization and used a drop of 0.2, the loss was stagnant.

Hence in our final model we used universal sentence encoder, which decreased the loss and model became realistic and efficient.

CONCLUSION

We mentioned before that if many of our modelling experiments are returning similar results, despite using different kinds of models, it's a good idea to return to the data and inspect why this might be.

One of the best ways to inspect your data is to sort your model's predictions and find the samples it got most wrong, meaning, what predictions had a high prediction probability but turned out to be wrong.

REFERENECES

1. Ashwin Rajadesingan, Reza Zafarani, and Huan Liu Computer Science and Engineering Arizona State University

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

3. Meishan Zhang1 , Yue Zhang2 and Guohong Fu1 1. School of Computer Science and Technology, Heilongjiang University, China 2. Singapore University of Technology and Design