SARCASM DETECTION

-By Hareetima Sonkar

Problem Description:

● Sarcasm is a complex sentiment expressed using intensified positive or positive words,

typically intended to convey a negative connotation.

● The challenge lies in interpreting sarcastic content correctly, especially in Natural

Language Processing (NLP) and sentiment analysis.

● Our goal is to build a model that can recognize and understand sarcastic behaviour and

patterns in text

1. Business Problem

Sarcasm Detection is a problem that needs to be tackled because in situations or businesses that are reliant on sentiment analysis, finding the difference can be crucial. Most messages can be misinterpreted, such as “Great! You have done a good job”.

Misinterpreting sarcasm can lead to inaccurate sentiment analysis, which can adversely affect businesses in various ways:

- Customer Feedback - misunderstanding sarcastic comments in reviews.

- Public Relations - Detecting sarcasm in public statements, news articles, and headlines more effectively. Understanding the public's true sentiment during a crisis to tailor responses that address genuine concerns rather than sarcastic criticisms.

- Brand Reputation Management - Brands may fail to identify and address negative sentiments disguised as sarcasm.

- Social Media Monitoring - Automated systems might misclassify sarcastic comments, leading to flawed insights and responses.

- Ad Campaign Analysis - evaluating the advertisement campaigns where consumer reactions might include sarcastic praise or criticism.

2. Solution Description

In this project, we will be building sarcasm detection using a neural network and classifying it as sarcasm or non-sarcasm. This implies that it is a classification problem.

Data Preprocessing

1. *Duplicate Removal*:

   - Identified and removed duplicate rows from the dataset.

2. *Data Summary:*

   - Provided a summary and concise information about the dataset, including value counts and basic statistics.

3. *Data Cleaning*:

   - Downloaded and utilized NLTK's stopwords.

   - Implemented a `clean_text` function to preprocess text data by lowering case, removing square brackets, and eliminating alphanumeric characters.

Exploratory Data Analysis

1. *Visualizations:*

- Generated visualizations like pie charts to show the distribution of labels (0s and 1s).

2. *Descriptive Statistics:*

- Displayed various descriptive statistics and data info to understand the dataset better.

Text Tokenization and Padding

- Utilized Keras' `Tokenizer` and `pad_sequences` for text tokenization and padding to prepare the text data for model training.

Model Training

1. *Model Architecture*:

- Defined and compiled a neural network model using Keras.

- The architecture included an embedding layer, LSTM, and Dense layers.

2. *Data Splitting*:

- Split the dataset into training and validation sets.

3. *Training:*

- Trained the model on the pre-processed data and evaluated its performance using accuracy and loss metrics.

Model Evaluation

1. *Performance Metrics:*

- Used confusion matrix and classification report to evaluate the model's performance on the test set.

2. *Visualizations*:

- Plotted training and validation accuracy and loss to visualize the model’s learning process.

Prediction

1. *Prediction Function*:

- Implemented a function to predict whether new sentences are sarcastic or not.

2. *Testing*:

- Tested the model on a list of input texts and displayed the predictions.

3. Dataset Description

The dataset contains text comments labelled as sarcastic or non-sarcastic. Each comment is a short piece of text sourced from social media or other platforms [Reddit].

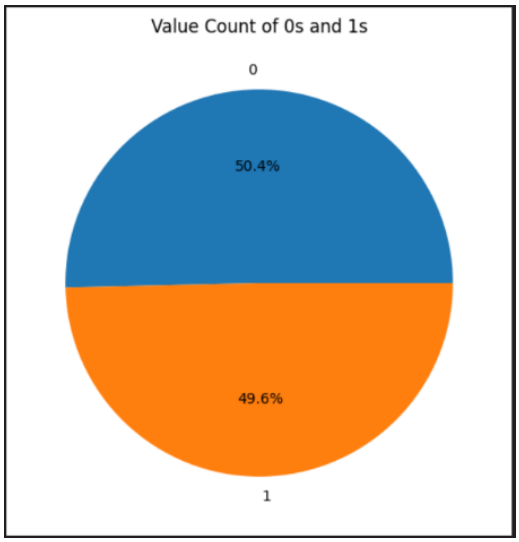
Dataset size: 1million

Examples:

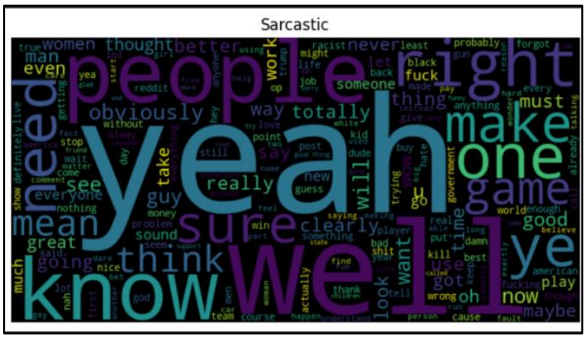
0	LOOOOOL brrraxxxxton	
0	Ahh did not know that I rarely use that playstyle so I was just going off spellbuff alone	
0	Hes becoming a mene like Dicks out for Harambe	
1	If it wasnt for the invention of the coin flip Avilo would never lose	
0	I laughed so hard at how fucking stupid that was	
1	If the officer carried an extra gun this wouldnt have happened	
1	If it doesnt do well in Japan it doesnt matter anywhere else	
0	miere si rahat	
1	Becuase Hillary is a beacon of honesty	
1	Just wait for wretched eye with void burn that will be fun	
0	You can with adequate regulation	
1	so youre saying technically pot is legal because what theyre regulating isnt actually the plant that people are interested in	
1	I havent heard that one before	
0	Im not sassing that man even via the interweb	
1	Troopers fucking unite	
1	Im sure you play multiple hours a week and commit to this game because you want a border and a victorious skin it sounds very legit and reasonnable	
0	That is not a comic book movie	
0	Usually most likely but it looks appropriate for a high5 too IMHO	
0	the salvation army	
1	Cicret has the ability to project dark light thats impressive and totally possible by the laws of physics	
0	Thats what I was thinking as well	

4. Data Visualization

- **Label-wise Split**: The dataset contains a balanced distribution of sarcastic and non-sarcastic comments, ensuring unbiased model training.



  - Sarcastic comments are visualized using a word cloud.



  - Non-sarcastic comments are visualized using a separate word cloud.

5. Data Preprocessing

Data preprocessing steps included:

- Converting text to lowercase.
- Removing text in square brackets and alphanumeric characters.
- Creating a new column to store cleaned text.
- Duplicate removing
- Null value removing

6. Tokenization and Embedding Techniques

Tokenization: The text data is tokenized using the Keras Tokenizer. The process involves:

- Fitting the tokenizer on the text data to create a word index.
- Converting each text sequence into a sequence of integers, where each integer represents a specific word in the word index.
- Padding the sequences to ensure they all have the same length, which is necessary for feeding the data into the neural network.

Embedding: The model uses an Embedding layer to convert tokens into dense vectors:

- The Embedding layer takes the integer-encoded sequences and maps each integer (word index) to a dense vector of fixed size (embedding_dim).
- The embedding weights are learned during the training process. These embeddings capture the semantic relationships between words, allowing the model to understand the context in which words appear.

7. Modelling

The primary model used is a deep-learning neural network. Here's a detailed explanation of the architecture:

- **Embedding Layer**: tf.keras.layers.Embedding(vocab_size, embedding_dim)
    - Converts word indices to dense vectors of a fixed size (embedding_dim). This layer learns an embedding for each word in the vocabulary during training.
- **Global Max Pooling Layer**: tf.keras.layers.GlobalMaxPool1D()
    - Reduces the dimensionality of the input by taking the maximum value over the time dimension for each feature map. This helps in extracting the most important features from the text.
- **Dense Layer 1**: tf.keras.layers.Dense(40, activation='relu')
    - A fully connected layer with 40 neurons and ReLU activation function. This layer learns complex features from the pooled embeddings.
- **Dropout Layer 1**: tf.keras.layers.Dropout(0.5)
    - A regularization technique where 50% of the neurons are randomly turned off during training to prevent overfitting.
- **Dense Layer 2**: tf.keras.layers.Dense(20, activation='relu')
    - A fully connected layer with 20 neurons and ReLU activation function. This layer continues learning more complex features.

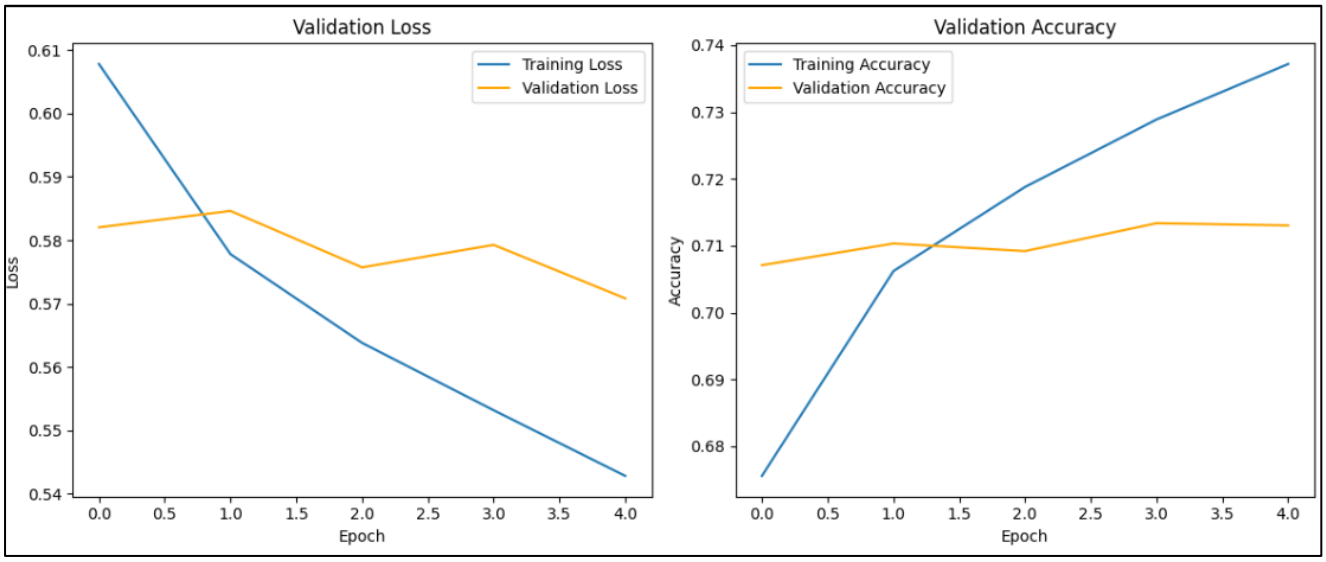
- **Dropout Layer 2**: tf.keras.layers.Dropout(0.5)
    - Another regularization layer with 50% dropout to prevent overfitting.
- **Dense Layer 3**: tf.keras.layers.Dense(10, activation='relu')
    - A fully connected layer with 10 neurons and ReLU activation function. This layer further refines the learned features.
- **Dropout Layer 3**: tf.keras.layers.Dropout(0.2)
    - A final regularization layer with 20% dropout to prevent overfitting.
- **Output Layer**: tf.keras.layers.Dense(1, activation='sigmoid')
    - A single neuron output layer with a sigmoid activation function, which outputs a probability score indicating the likelihood of the input text being sarcastic.
- **Training**: The model is trained on the training dataset and validated on the validation dataset.

```
Epoch 1/5
21756/21756 ━━━━━━━━━━━━━━━━━━━━ 344s 16ms/step - accuracy: 0.6472 - loss: 0.6275 - val_accuracy: 0.7071 - val_loss: 0.5820
Epoch 2/5
21756/21756 ━━━━━━━━━━━━━━━━━━━━ 345s 16ms/step - accuracy: 0.7068 - loss: 0.5773 - val_accuracy: 0.7103 - val_loss: 0.5846
Epoch 3/5
21756/21756 ━━━━━━━━━━━━━━━━━━━━ 342s 16ms/step - accuracy: 0.7210 - loss: 0.5615 - val_accuracy: 0.7092 - val_loss: 0.5757
Epoch 4/5
21756/21756 ━━━━━━━━━━━━━━━━━━━━ 344s 16ms/step - accuracy: 0.7314 - loss: 0.5500 - val_accuracy: 0.7134 - val_loss: 0.5793
Epoch 5/5
21756/21756 ━━━━━━━━━━━━━━━━━━━━ 350s 16ms/step - accuracy: 0.7398 - loss: 0.5395 - val_accuracy: 0.7130 - val_loss: 0.5708

Model: "sequential"
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 60, 200)	2,000,000
global_max_pooling1d (GlobalMaxPooling1D)	(None, 200)	0
dense (Dense)	(None, 40)	8,040
dropout (Dropout)	(None, 40)	0
dense_1 (Dense)	(None, 20)	820
dropout_1 (Dropout)	(None, 20)	0
dense_2 (Dense)	(None, 10)	210
dropout_2 (Dropout)	(None, 10)	0
dense_3 (Dense)	(None, 1)	11

- **Evaluation**: The model's performance is evaluated on the test dataset.

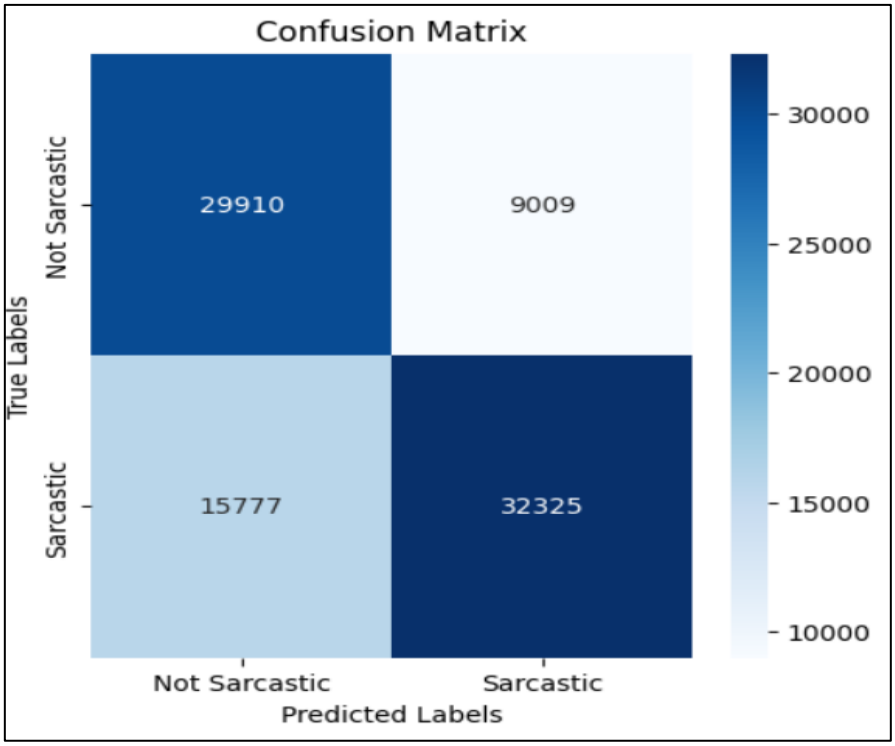


8. Evaluation Metrics

Key evaluation metrics included:

- **Accuracy**: Overall correctness of the model predictions.
- **Precision**: True positive rate among predicted positives.
- **Recall**: True positive rate among actual positives.
- **F1-Score**: Harmonic mean of precision and recall.
- **Best Model**: The deep learning model achieved an accuracy of 72%, indicating robust performance in sarcasm detection.

Confusion Matrix:



True Positives (TP)

- Count: 32,325
- Interpretation: The model correctly identified 32,325 instances as Sarcastic.

True Negatives (TN)

- Count: 29,910
- Interpretation: The model correctly identified 29,910 instances as Not Sarcastic.

False Positives (FP)

- Count: 15,777
- Interpretation: The model incorrectly identified 15,777 instances as Sarcastic when they were actually Not Sarcastic.

False Negatives (FN)

- Count: 9,009
- Interpretation: The model incorrectly identified 9,009 instances as Not Sarcastic when they were actually Sarcastic.

Classification Report:

Classification Report:	precision	recall	f1-score	support
Not Sarcastic	0.65	0.77	0.71	38919
Sarcastic	0.78	0.67	0.72	48102
accuracy			0.72	87021
macro avg	0.72	0.72	0.71	87021
weighted avg	0.73	0.72	0.72	87021

The classification report provides a detailed summary of various metrics for each class (Not Sarcastic and Sarcastic) and overall metrics for the model's performance. Here's a breakdown of each section:

• Precision

Precision measures the proportion of positive identifications (in this case, predictions of Sarcastic) that were actually correct.

Precision for Not Sarcastic: 0.65

Interpretation: When the model predicted an instance as Not Sarcastic, it was correct 65% of the time.

Precision for Sarcastic: 0.78

Interpretation: When the model predicted an instance as Sarcastic, it was correct 78% of the time.

• Recall (Sensitivity)

Recall measures the proportion of actual positives (in this case, instances that are actually Sarcastic) that were correctly identified by the model.

Recall for Not Sarcastic: 0.77

Interpretation: The model identified 77% of all actual Not Sarcastic instances correctly.

Recall for Sarcastic: 0.67

Interpretation: The model identified 67% of all actual Sarcastic instances correctly.

•F1-Score

The F1-score is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall.

F1-score for Not Sarcastic: 0.71

Interpretation: The harmonic mean of precision (0.65) and recall (0.77) for Not Sarcastic.

F1-score for Sarcastic: 0.72

Interpretation: The harmonic mean of precision (0.78) and recall (0.67) for Sarcastic.

•Support

Support is the number of actual occurrences of each class in the dataset.

Support for Not Sarcastic: 38,919

Support for Sarcastic: 48,102

•Accuracy

Accuracy measures the overall correctness of the model across all classes.

Accuracy: 0.72 (or 72%)

Interpretation: The model correctly predicted **72%** of all instances.

Final Outputs/Predictions:

```
Enter a sentence for prediction (or type 'exit' to quit):  Oh great, another meeting that could have been an email.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
sentence: Oh great, another meeting that could have been an email.
Prediction: Sarcastic
Enter a sentence for prediction (or type 'exit' to quit):  The weather has been beautiful this week.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
sentence: The weather has been beautiful this week.
Prediction: Not Sarcastic
Enter a sentence for prediction (or type 'exit' to quit):  Thanks for helping me with the presentation.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 993us/step
sentence: Thanks for helping me with the presentation.
Prediction: Not Sarcastic
Enter a sentence for prediction (or type 'exit' to quit):  I appreciate all the hard work you put into this project.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step
sentence: I appreciate all the hard work you put into this project.
Prediction: Sarcastic
Enter a sentence for prediction (or type 'exit' to quit):  I enjoyed the movie we watched last night.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
sentence: I enjoyed the movie we watched last night.
Prediction: Not Sarcastic
Enter a sentence for prediction (or type 'exit' to quit):  I just love when people don't use their turn signals.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
sentence: I just love when people don't use their turn signals.
Prediction: Sarcastic
Enter a sentence for prediction (or type 'exit' to quit):  Just what I needed, more work to do over the weekend.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
sentence: Just what I needed, more work to do over the weekend.
Prediction: Not Sarcastic
```