Contextual Sarcasm Detection on Social Media Comments

Team Members

- Sanath Kumar Vobilisetty
- Venkata Rakesh Kumar Molakala

Introduction

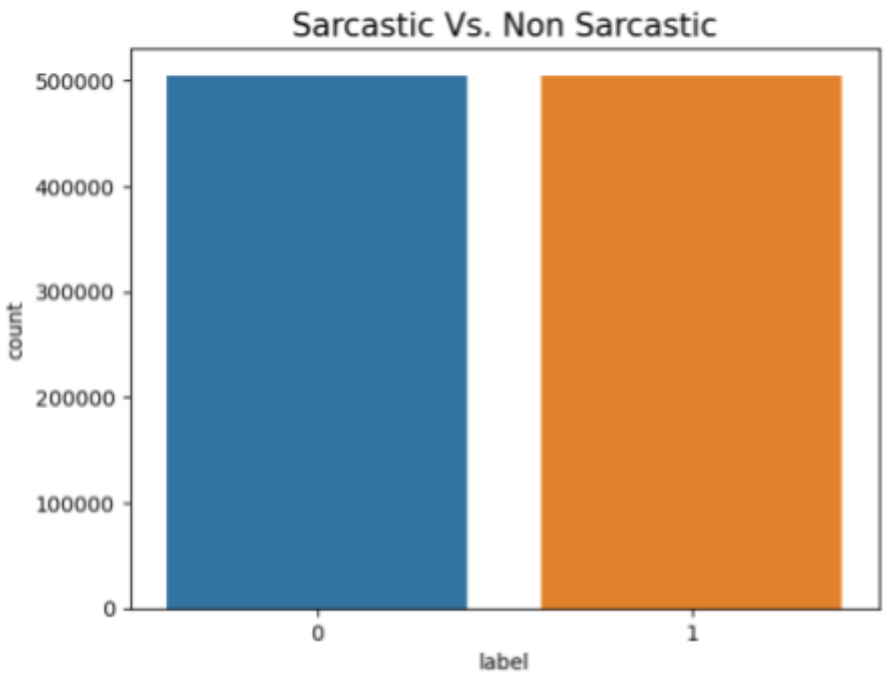
Sarcasm is a form of language used to express contempt, ridicule or irony by stating the opposite of what is actually meant. It is a commonly used form of communication in social media platforms, such as Reddit, where people engage in discussions and express their opinions. However, detecting sarcasm in text is not an easy task for machines due to the complexity of language and the various ways in which sarcasm can be expressed.

Natural Language Processing (NLP) models have been developed to solve this problem, using machine learning algorithms to detect sarcasm in text data. In this project, we aim to build a sarcasm detection NLP model using a dataset of Reddit comments, and compare the performance of various models such as Naive Bayes, Logistic Regression, CNN, and LSTM to determine the best approach. The results of this project can have practical applications in various domains, such as sentiment analysis, customer service, and social media monitoring.

Dataset

Dataset on Kaggle
The data utilized comprises 1.3 million humorous remarks sourced from Reddit, a website for online discussions. The comments were collected through web scraping from posts that included the \s (sarcasm) tag. This particular tag is commonly employed by Redditors to signal that their remark is intended as a joke and should not be interpreted literally. As a rule of thumb, the presence of this tag indicates that the comment is sarcastic.

- Dataset has both balanced and imbalanced versions and has a size of ~243MB.
- The dataset consists of 14 different columns out of which the most significant ones we considered are:
    - Label (Integer): Predictor columns that indicate sarcastic/not sarcastic.
    - Comment (String): Reply to a Parent Reddit comment
    - Subreddit (String): Commented under which subreddit
    - Score (Integer): Number of upvotes -(minus) Number of downvotes.



Number of Sarcastic and Non Sarcastic comments plot

Data preprocessing

To build a proper NLP model for sarcasm detection, preprocessing of the comment column in the dataset is crucial. The preprocessing steps include removing punctuations, converting all text to lowercase, removing stopwords, and decontracting words.

- Removing punctuations and converting all text to lowercase helps to standardize the text data, making it easier to analyze and remove noise from the dataset.

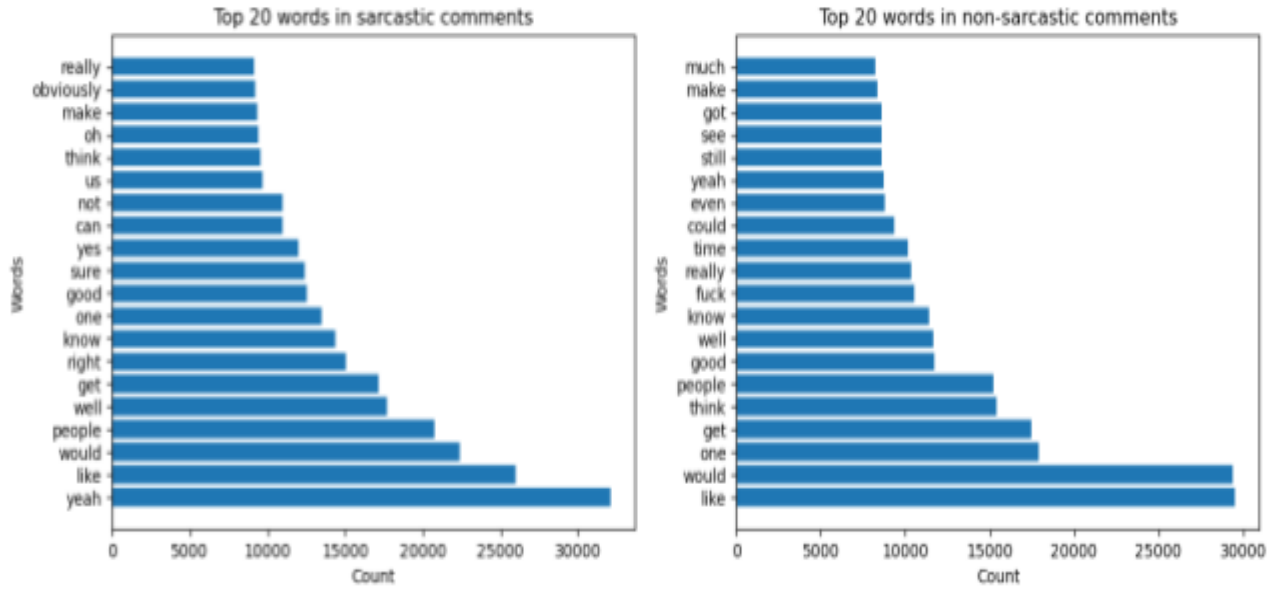
- Removing stopwords such as "and", "the", "a", etc., is also important as they do not carry any sentiment and may introduce noise to the dataset.

- Decontracting words, such as "don't" to "do not" and "can't" to "can not", is also necessary because contractions may lead to ambiguity in sentiment analysis. Expanding contractions to their full form makes the text more explicit and removes ambiguity, resulting in better sentiment analysis results.

Preprocessing the dataset in this way helps to improve the accuracy and reliability of sentiment analysis for sarcasm detection. By standardizing the text data and removing noise, the model can more accurately identify sentiment-bearing words that are crucial for sarcasm detection. Additionally, the model is better able to distinguish between sarcastic and non-sarcastic comments, resulting in more accurate and reliable predictions.

Data Visualization

The top 20 words in sarcastic comments and non-sarcastic comments can provide valuable insights into the language used in each type of comment.

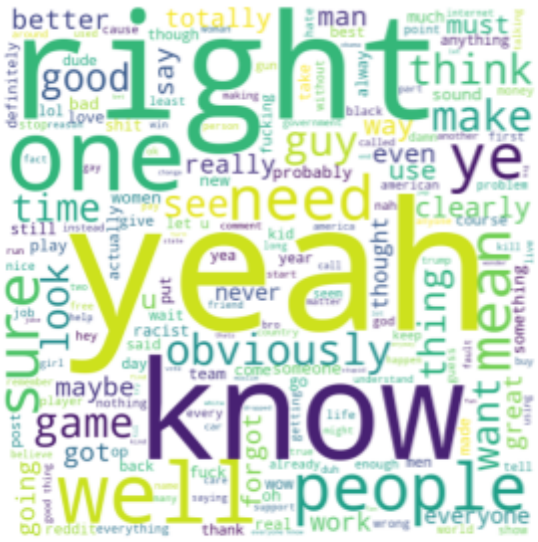


From the top 20 words in sarcastic comments, we can see that words such as "really", "obviously", and "not" appear frequently. This suggests that sarcasm often involves the use of words that contradict or undermine the meaning of the text, and that sarcasm is often used to express opinions that are different from what is expected or assumed. Additionally, words such as "good", "well", and "like" also appear frequently in sarcastic comments, suggesting that sarcasm can also involve a sense of irony or humour.

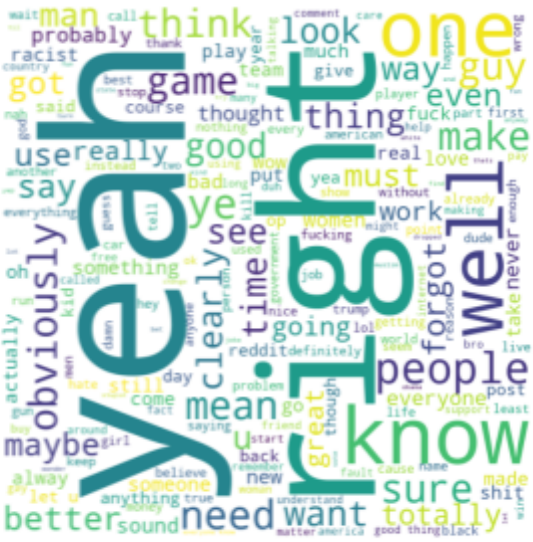
On the other hand, the top 20 words in non-sarcastic comments reveal a different set of commonly used words. Words such as "much", "make", and "got" appear frequently, suggesting that non-sarcastic comments may be more focused on providing information or discussing topics in a straightforward manner.

Overall, the top 20 words in sarcastic and non-sarcastic comments provide insight into the language used in each type of comment. This information can be useful in developing an NLP model for sarcasm detection, as it can help to identify common linguistic features of sarcastic comments that can be used to train the model.

Similarly we can observe the supporting information in the WordCloud of both Sarcastic comment words and non sarcastic comment words. As we can see, there is no specific pattern that allows us to determine whether a particular word is sarcastic or not. Therefore, we cannot rely solely on the presence of words in a sentence to determine whether a comment is sarcastic or not.



WordCloud of Sarcastic comments



WordCloud of Non Sarcastic comments

Data Modelling

We then split our dataset into training and testing sets using the train_test_split function from the sklearn library. We chose a 80-20 split as it is a commonly used split ratio and can provide a balance between having enough data to train the model while still having a sufficient amount of data to evaluate its performance. We have considered the following models and compared their performance:

1. **Naive Bayes(Bag of Words)**

   We used the Bag of Words approach to represent our textual data in a numerical format that could be used as input to our machine learning model. The CountVectorizer function was applied to our training data to create a numerical representation of the text, which was then used to train our model to detect sarcasm in social media comments.

   Similarly, the CountVectorizer function was applied to our testing data to transform it into a numerical format, and the resulting features were stored in the test_features variable. We then used the Naive Bayes algorithm as our machine learning model for sarcasm detection and created an instance of the Naive Bayes classifier, which was then trained on our training data using the bag of words features.

2. **Logistic Regression**

   To further improve the accuracy of our sarcasm detection model, we used the TfidfVectorizer approach. Our training and testing data was transformed into a numerical format using the TfidfVectorizer function.

   We then used the logistic regression algorithm to train our sarcasm detection model on the TfidfVectorizer features. The classifier was then trained on the training data using the TfidfVectorizer features. By using the TfidfVectorizer approach with logistic

regression, we were able to further improve the accuracy of our sarcasm detection model.

3. **CNN**

   We have first preprocessed the comments by tokenizing them using the Keras tokenizer and padding the sequences to a fixed length of 100. This is necessary for feeding the comments into the CNN model.

   We then built a CNN model with multiple layers including an Embedding layer, Convolutional layer, MaxPooling layer, Dense layer and Dropout layer. The CNN mode is trained using the preprocessed data and validated on a test set. The model was trained for 10 epochs with a batch size of 32, using binary crossentropy loss and the Adam optimizer. The performance of the CNN model is then evaluated based on various metrics such as accuracy, precision, recall, F1-score, etc.

4. **LSTM**

   In addition to the previous models, we also implemented a Long Short-Term Memory (LSTM) model for sarcasm detection in social media comments. LSTMs are a type of neural network architecture that are particularly useful for processing sequential data.

   We used an embedding layer with an input dimension of 10,000, output dimension of 128, and input length of 100 to preprocess our text data. The embedding layer maps each word to a dense vector, which helps capture the semantic meaning of words and their relationships with other words in the text.

   We then added an LSTM layer with 64 units, and applied dropout regularization to prevent overfitting. Finally, we added a dense layer with a sigmoid activation function to predict the output class (sarcastic or non-sarcastic). We compiled the model with binary cross-entropy loss and accuracy metrics, and trained it using our preprocessed data. We finally evaluated the model by computing the loss and accuracy on the training set.

5. **Bi-LSTM**

   After the LSTM model, we even worked on building a Bi-LSTM model. The model architecture consists of an embedding layer, followed by a bidirectional LSTM layer, a dropout layer to prevent overfitting, and a dense output layer with sigmoid activation.

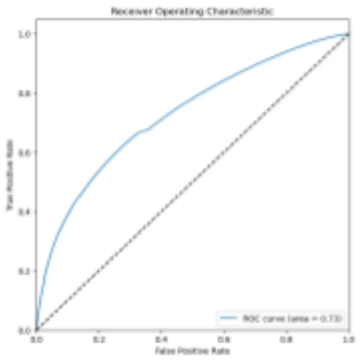
   During training, the model is compiled with binary cross-entropy loss and the Adam optimizer. The model is trained for 5 epochs with a batch size of 32 and validated on a separate validation set.

   Overall, the bidirectional LSTM model is a popular approach for natural language processing tasks such as sentiment analysis and sarcasm detection. By using a bidirectional LSTM, the model is able to capture both past and future context when making predictions. The dropout layer helps to prevent overfitting, and the use of binary cross-entropy loss and sigmoid activation in the output layer is appropriate for binary classification tasks like sarcasm detection.

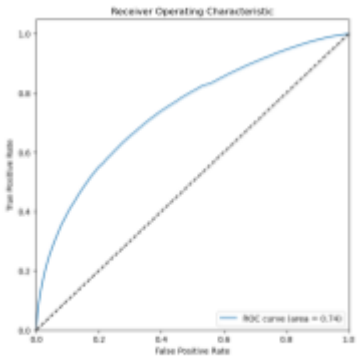
Performance Evaluation

Model	Test Accuracy	Precision (1 - sarcastic)	Recall (1 - sarcastic)	F1-score (1 - sarcastic)
Naive Bayes (Bag of words)	0.659	0.65	0.68	0.66
Naive Bayes (TFIDF)	0.656	0.65	0.67	0.66
Logistic Regression	0.679	0.69	0.64	0.67
CNN	0.658	0.66	0.66	0.66
LSTM	**0.770**	**0.80**	**0.73**	**0.76**
Bi-LSTM	0.687	0.71	0.63	0.67

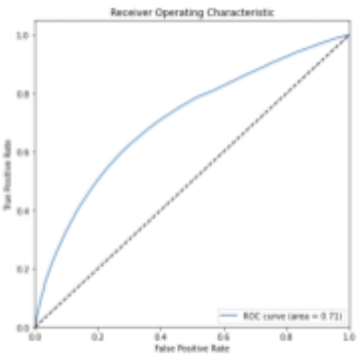
ROC Curves of respective models:



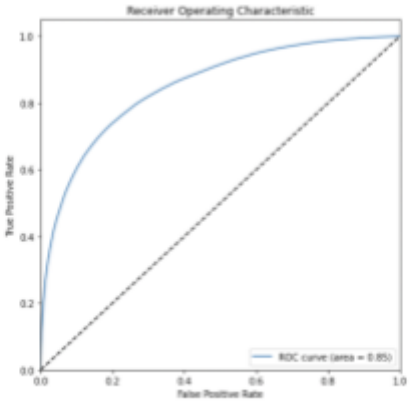
Naive Bayes (bow)



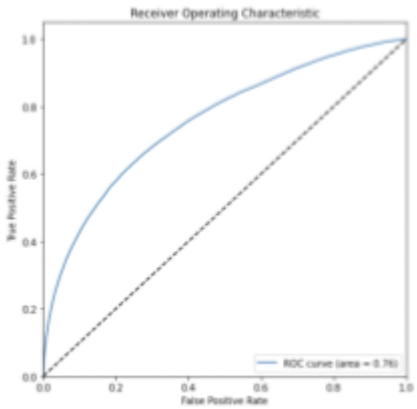
LOGIT



CNN



LSTM



Bi-LSTM

- Naive Bayes (Bag of words) and Naive Bayes (TFIDF) have similar accuracy, recall, and F1-score, but Bag of words outperforms TFIDF in terms of precision.
- Logistic Regression has better accuracy than Naive Bayes models, but its precision and recall scores are comparatively lower.
- CNN has an accuracy similar to the Naive Bayes models but has a balanced precision and recall score, leading to an average F1-score.
- LSTM has the highest accuracy among all models, and it has the best precision score, which means it is more effective in identifying sarcastic comments. It also has a good recall score, indicating that it is not missing many sarcastic comments. The F1-score of LSTM is also high, suggesting that it is a well-balanced model.
- Bi-LSTM has lower accuracy, precision, and recall scores than LSTM but still performs better than the other models except LSTM.

Ablation Study

To perform ablation study in our project, we can systematically remove or disable different layers or components of each model and evaluate the model's performance with and without that layer or component.

Model	Current Accuracy	Ablation Parameters Considered	Post Ablation Accuracy
Naive Bayes	0.659	● Stemming ● Alpha Smoothing	0.652
Logit	0.679	● Stemming ● Regularization parameter	0.672
CNN	0.658	● Stemming ● Size of Convolution filters	0.656
LSTM	0.770	● Stemming ● Number of LSTM layers ● Type of attention mechanism	0.740
Bi-LSTM	0.687	● Stemming ● Dropout rate applied to hidden states	0.690

- Based on the results of your ablation study, it can be concluded that the Naive Bayes and Logit models are more sensitive to changes in the selected parameters compared to CNN, LSTM, and Bi-LSTM models.
- For Naive Bayes, the post-ablation accuracy is lower than the initial accuracy, indicating that the removed parameters (stemming and alpha smoothing) were contributing positively to the model's performance.
- On the other hand, for Logit, the post-ablation accuracy is slightly lower than the initial accuracy, indicating that the removed parameters (stemming and regularization parameter) had a positive impact on the model's performance, but not as significant as in the case of Naive Bayes.

- For the CNN model, the post-ablation accuracy is slightly lower than the initial accuracy, suggesting that the removed parameter (size of convolution filters) contributed positively to the model's performance, but the impact was not significant.
- For LSTM, the post-ablation accuracy is significantly lower than the initial accuracy, indicating that the removed parameters (number of LSTM layers and attention mechanism) were contributing positively to the model's performance.
- For Bi-LSTM, the post-ablation accuracy is almost the same as the initial accuracy, indicating that the removed parameters (stemming and dropout rate) did not have a significant impact on the model's performance.
- Overall, the ablation study helped to identify the critical parameters that affect the performance of the models and provided insights into the strengths and weaknesses of each model.
- The critical parameters may include the number of LSTM layers, the type of attention mechanism, the size of convolution filters, the dropout rate applied to hidden states, and the regularization parameter in logistic regression.

Conclusion and Future work

In conclusion, we have implemented various machine learning and deep learning models to perform sarcasm detection in social media comments. We have analysed the performance of each model using different evaluation metrics and performed ablation studies to identify the critical parameters that contribute to the performance of the models.

Based on the results obtained, we can conclude that the LSTM model outperforms all other models with an accuracy of 0.77, precision of 0.80, recall of 0.73, and F1-score of 0.76. The ablation study also confirmed that stemming is a critical parameter that contributes significantly to the performance of the LSTM model.

In the future, we can explore various other techniques to improve the performance of sarcasm detection models. One potential direction is to use more advanced attention mechanisms to better capture the complex relationships between the words in the text. Additionally, we can explore the use of pre-trained language models such as BERT or GPT to improve the model's performance. Another potential area of exploration is to incorporate contextual information such as user profiles, emotions, and topic domains, which can help to improve the model's accuracy and interpretability.

Overall, the results of this project demonstrate the potential of deep learning models for sarcasm detection in social media comments and provide a foundation for future research in this field.

References

- Jena, Amit Kumar, Aman Sinha, and Rohit Agarwal. "C-net: Contextual network for sarcasm detection." Proceedings of the second workshop on figurative language processing. 2020.

- Yaghoobian, Hamed, Hamid R. Arabnia, and Khaled Rasheed. "Sarcasm detection: A comparative study." arXiv preprint arXiv:2107.02276 (2021).

- Du, Yu, et al. "An effective sarcasm detection approach based on sentimental context and individual expression habits." Cognitive Computation (2022): 1-13.

- Das, Dipto, and Anthony J. Clark. "Sarcasm detection on facebook: A supervised learning approach." Proceedings of the 20th international conference on multimodal interaction: adjunct. 2018.

- Bouazizi, Mondher, and Tomoaki Otsuki Ohtsuki. "A pattern-based approach for sarcasm detection on twitter." IEEE Access 4 (2016): 5477-5488.