Final Report
Analyzing User Behavior, Trends and Communities using Reddit Posts
Sentiment Analysis of Subreddits

***Abstract* – This study explores sentiment analysis on subreddit comments to understand if certain topics elicit more positive or negative sentiment. Challenges include sarcasm, GIFs(Visual content), diverse topics, multimodal content, and informal nature of texts. We employ transformer models like BERT, RoBERTa, DistilBERT and XLNet for analysis, aiming to enhance understanding of online discourse dynamics. Future work will address visual data handling using Image processing to extract a short description of the content to check relevancy of comment.**

I INTRODUCTION

This research delves into the realm of sentiment analysis within the diverse and dynamic landscape of Reddit, a popular online platform teeming with user-generated content. Our study commenced with the identification of trending topics across a broad spectrum of categories such as Education, Sports, Entertainment, Food, Companies, Politics, Nature, Memes, Technology, and Gaming through trend analysis. To gain a deeper understanding of these topics, we zeroed in on the most influential subreddits within each category. Subreddits, the smaller communities within Reddit, serve as concentrated hubs for users to share and discuss content related to a specific topic. By focusing on these subreddits, we were able to amass a rich and diverse dataset for our analysis.

Our methodology involves basic preprocessing of the data, which includes handling gifs, emojis, emoticons, and most importantly, sarcasm. We initially employ LSTM as a baseline model for our sentiment analysis. Subsequently, we leverage state-of-the-art transformer models such as BERT, RoBERTa, DistilBERT, and XLNet to enhance the accuracy of our sentiment analysis.

RESEARCH GOAL: This research is to test the hypothesis of whether certain communities elicit more positive or negative sentiments. Through this endeavor, we aim to shed light on the sentiment dynamics within various online communities, thereby contributing to the broader understanding of online discourse.

II LITERATURE REVIEW

2.1 Sentiment Analysis of AMC, Clover, and GameStop Reddit Comments[1]:

Summary: The research utilizes the Clov package in R Studio and the NRC lexicon for sentiment analysis on Reddit comments about AMC, Clover, and GameStop. Network diagrams and radar charts present nuanced insights into distinct sentiment trends for each product. The study combines network and sentiment analysis, offering a holistic view of user interactions.

Drawbacks: However, limitations include reliance on a snapshot of data, potentially missing real-time sentiments, and insufficient exploration of scalability for larger datasets or multiple subreddits.

2.2 Enhancing Tweet Sentiment Analysis through Feature Ensemble Modeling and CNN[2]:

Summary: The paper addresses limitations in existing sentiment analysis methods by introducing a novel feature ensemble model. This approach considers lexical, word-type, semantic, position, and sentiment polarity features in tweets containing fuzzy sentiment. Evaluation with real data, including DB1 and DB2 datasets, demonstrates enhanced sentiment analysis performance, notably in F1 scores.

Drawbacks: the study acknowledges challenges related to the interpretability of deep learning models, especially CNNs, urging attention to the transparency of decision-making processes.

2.3 Sentiment Analysis using RGWE embeddings with improved accuracy[3]:

Summary: The paper introduces Refined Global Word Embeddings (RGWE) as a superior method for sentiment analysis, demonstrating its effectiveness across various datasets, experiment settings, and factors like sentiment concept weight and semantic similarity threshold. RGWE, incorporating position features and internal/external sentiment information, outperforms traditional embeddings in capturing sentiment nuances.

Drawbacks: The study acknowledges limitations like optimization for specific datasets and a lack of discussion on the method's generalizability.

2.4 Sentiment analysis for measuring hope and fear from Reddit posts during the 2022 Russo-Ukrainian conflict[4]:

Summary : The paper introduces an unsupervised sentiment analysis method to measure “hope” and “fear” sentiments in the context of the 2022 Ukrainian-Russian Conflict. Reddit data from six relevant subreddits was extracted using a Python script. The data was analyzed to gauge interest in the conflict and count “hope” and “fear” word occurrences. Topic modeling was also performed on the text data, with optimal topics estimated and goodness-of-fit measures along with Word clouds This research provides a novel sentiment analysis approach for geopolitical conflicts.

2.5 A survey on sentiment analysis methods, applications, and challenges[5]:

Summary: The paper discusses sentiment analysis data collection methods, including web scraping, social media, news channels, E-commerce websites, forums, and other websites. It highlights the importance of feature identification for model development and discusses the use of "Uni-gram", "Bi-gram", and "Tri-gram" techniques. The paper also mentions pragmatic features that focus on word application. Feature extraction is emphasized as a crucial task in sentiment classification. The paper explores three sentiment analysis approaches: the Lexicon Based Approach, the Machine Learning Approach, and the Hybrid Approach.

III DATASET

3.1 Dataset Selection and Extraction:

To align with the research goal of analyzing sentiment in online discussions across various domains, we meticulously selected existing datasets covering diverse topics such as Movies, Sports, Politics, Education, Science and Technology, Literature, Agriculture, etc. This selection ensures a comprehensive exploration of sentiment dynamics across different subjects. For each topic, we chose 3 subreddits with highest members, 3 with highest posts, and top 3 trending subreddits. If there are common names in this, we take the next one.

3.1.1 Data Schema Definition:

In order to facilitate effective merging and analysis of datasets, we extracted data using the Python Reddit API Wrapper (PRAW) to ensure compatibility with existing datasets. This step is critical for creating a unified schema. The dataset comprises essential information, including comments and other relevant details, sourced from a multitude of subreddits.

3.1.2 Reddit Data Extraction:

Employing PRAW, we systematically extracted data from Reddit, focusing on predetermined subreddits, types of posts, and pertinent parameters aligned with our objective of emotion mining. The information retrieved encompasses post titles, content, comments, scores, and other metadata crucial for our analysis.

3.1.3 Dataset Merging:

To streamline our analysis, we merged datasets obtained through PRAW, utilizing common identifiers. This merging process ensures that our dataset maintains a unified schema, a prerequisite for a coherent and meaningful analysis across diverse topics and domains.

3.2 Dataset Overview:

The dataset comprises the following columns:
'comment_created_utc': Timestamp of comment creation
'category': Topic category of the subreddit
'subreddit': Subreddit to which the comment belongs
'post_id': Identifier for the post associated with the comment
'comment_id': Unique identifier for the comment
'comment_body': Text content of the comment
'comment_ups': Number of upvotes received by the comment
'comment_downs': Number of downvotes received by the comment(if negative ups)
'comment_author': Author of the comment
sentiment_score: Score indicating the sentiment of the comment
'is_sarcastic': Binary indicator for sarcasm (1 for sarcasm, -1 for non-sarcasm and 0 for neutral)
'probability_sarcastic': Probability score of the comment being sarcastic
'sentiment': The label given to the comment(positive)
This dataset structure encompasses temporal information, comment details, sentiment scores, and indicators for sarcasm, forming the foundation for our comprehensive analysis of sentiment dynamics across diverse subreddits and topics.

IV DATA PREPROCESSING

The data preprocessing stage involved several steps to clean and prepare the data for further analysis.

4.1 Text Cleaning: The first step was text cleaning, which involved removing non-word characters from the text. This was done to ensure that the text data was free of any special characters or symbols that could interfere with the analysis.

4.2 Case Normalization: The next step was case normalization, which involved converting all the text to lower case. This was done to ensure consistency in the data and to prevent the same words in different cases from being treated as different words.

4.3 Tokenization: After cleaning the text and normalizing the case, the text was tokenized. This involved splitting the text into individual words or tokens. This is a crucial step in text analysis as it allows us to analyze the text at the word level.

4.4 Stopword Removal and Lemmatization: The next step involved removing stopwords and lemmatizing the tokens. Stopwords are common words like 'is', 'the', 'and', etc., that do not carry much meaning and are often removed in text analysis. Lemmatization is the process of reducing a word to its base or root form. For example, the words 'running', 'runs', and 'ran' would all be reduced to the base form 'run'. This helps in reducing the dimensionality of the data and makes the analysis more efficient.

4.5 Handling Missing Values: The dataset was then checked for missing values. Any rows with missing values in the 'comment_body' column were dropped. Alternatively, missing values could be filled with an empty string or a string of your choice. This ensures that the dataset is complete and ready for analysis.

4.6 Handling Emojis and Emoticons: The text data was then processed to handle emojis and emoticons. Emojis were converted to words, and the colons around the word representation of the emoji were removed. Emoticons, which are text representations of facial expressions, were replaced with their meanings. This helps in capturing the sentiment expressed through emojis and emoticons.

4.7 Handling GIFs: Finally, the comments were processed to handle GIFs. Any URLs ending in .gif were identified, and the GIFs were downloaded and converted into a sequence of images. This was done to allow for the analysis of the visual content in the comments.

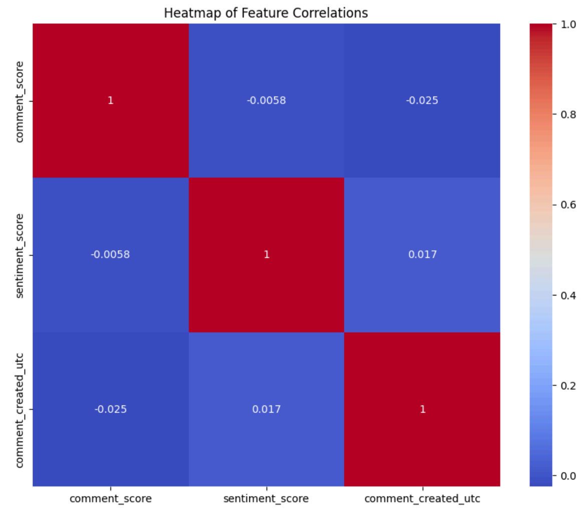


Fig 1: Correlation Matrix for features after preprocessing

Based on the correlation values, it appears that there is very little to no linear correlation between the pairs of features. This means that changes in one feature do not linearly correspond to changes in another feature.

4.8 Independence of Features: The features are largely independent of each other. Logistic regression assumes that the input features are independent of each other, so it can be used as a baseline model.

V HANDLING SARCASM IN SENTIMENT ANALYSIS

In the project's next stage, we addressed the issue of detecting sarcastic comments in our dataset. Sarcasm can reverse a statement's sentiment, complicating sentiment analysis models' predictions. We devised a technique to identify sarcasm in comments and adjust their sentiment scores.

5.1 Training a Model on Sarcastic Reddit Comments

We initiated our process by training a Naive Bayes model, a common choice for text classification, on a distinct dataset[8] filled with Reddit comments labeled as sarcastic or non-sarcastic. To ready the text data for the model, we employed a TF-IDF vectorizer. This tool transforms the text into a matrix of TF-IDF features, with TF-IDF standing for term frequency-inverse document frequency, a numerical statistic indicating a word's importance to a document in a corpus. After partitioning the data into training and test subsets, we trained the Naive Bayes model using the training data and subsequently assessed its accuracy with the test data.

5.2 Predicting Sarcasm in the Original Dataset

With the model trained, we then used it to predict whether the comments in our original dataset were sarcastic or not. We transformed the comments in the original dataset using the same TF-IDF vectorizer and made predictions using the trained model. The predictions were added as a new column in the original DataFrame, effectively labeling each comment as either sarcastic or non-sarcastic. In addition to the predictions, we also calculated the predicted probabilities of each comment

5.3 Adjusting Sentiment Scores

The final step was to adjust the sentiment scores of the comments based on whether they were predicted to be sarcastic or not. For each comment that was predicted to be sarcastic, we inverted its sentiment score. For example, the statement "I just love getting stuck in traffic" is likely to be sarcastic, and its true sentiment is negative, not positive. The Fig

2 below suggests the average sentiment reduces for all topics after we consider sarcastic comments.

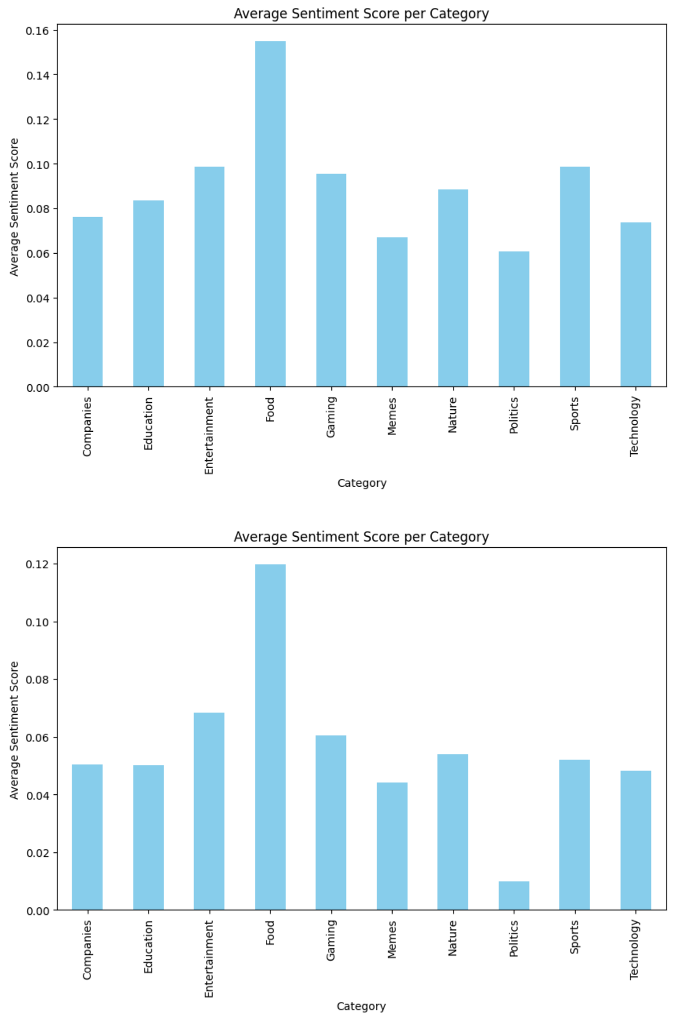


Fig 2: Compare avg sentiment scores before and after checking for sarcasm

Algorithm 1 Sarcasm Detection and Sentiment Update

```
Require: df_sarcastic (sarcasm dataset), df (original dataset)
Ensure: df (updated original dataset with sarcasm predictions and updated
    sentiment scores)
 1: df_sarcastic ← LOAD("path_to_sarcasm_dataset.csv")
 2: comments, labels ← PREPROCESS(df_sarcastic)
 3: comments_train, comments_test, labels_train, labels_test ←
    SPLIT(comments, labels, test_size=0.2)
 4: vectorizer ← INITIALIZE_TFIDF()
 5: comments_train_tfidf, comments_test_tfidf ← TRANSFORM(comments_train, comments_test, vectorizer)
 6: model ← TRAIN_MODEL(comments_train_tfidf, labels_train)
 7: accuracy ← EVALUATE_MODEL(model, comments_test_tfidf, labels_test)
 8: comments_original_tfidf ← TRANSFORM(df['comment_body'], vectorizer)
 9: df['is_sarcastic'] ← PREDICT(model, comments_original_tfidf)
10: df['probability_sarcastic'] ← PREDICT_PROBA(model, comments_original_tfidf)
11: df ← UPDATE_SENTIMENT(df) return df
```

The formula for Naive Bayes used:

$$P(\text{Sarcastic}|d) = \frac{P(\text{Sarcastic}) \cdot P(w_1|\text{Sarcastic}) \cdot P(w_2|\text{Sarcastic}) \cdot \ldots \cdot P(w_n|\text{Sarcastic})}{P(d)}$$

P(Sarcastic | d) is the probability that a comment d is sarcastic.
P(Sarcastic) is the prior probability of a comment being sarcastic.
P(wi | Sarcastic) is the probability of word wi appearing in a sarcastic comment.
P(d) is the prior probability of the comment.

Fig 3 shows sarcasm's prevalence in various subreddits, with a higher incidence in 'politics', 'worldnews', 'news', 'funny', 'gifs', 'pics', 'aww', 'gaming', 'videos', and 'todayilearned'. These communities often use sarcasm, possibly for humor or critique. The context and culture of each subreddit can influence this. The study emphasizes sarcasm's role in online discourse and its significance in community interaction and engagement, particularly in Politics, Memes, and Gaming subreddits

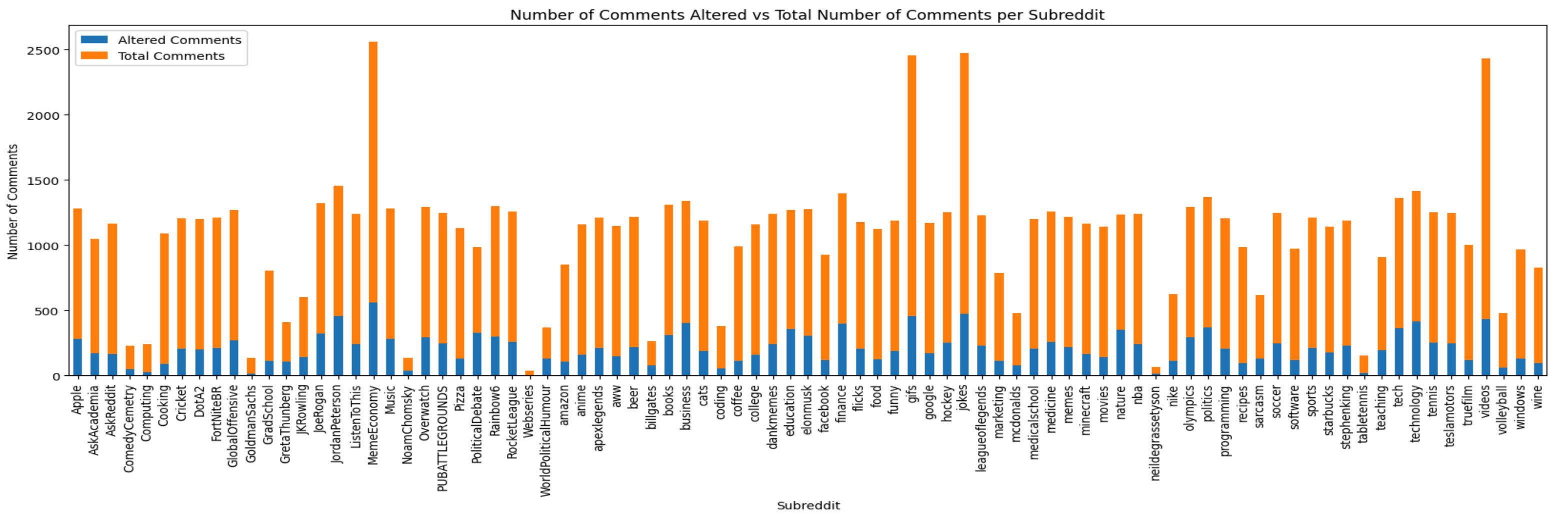


Fig 3: Analyzing the the number of comments altered for various subreddits

VI MODELING

6.1 Baseline Model:

6.1.1 Data Preparation:

We started by encoding our target variable, using a LabelEncoder. This converted our categorical target variable into a format that could be used with our LSTM model. Next, we tokenized our text data using Keras's Tokenizer class. We set a vocabulary size of 10,000 words and used the '<OOV>' token to represent words that were not in our vocabulary. We then converted our text data into sequences of tokens and padded these sequences to a consistent length of 128 tokens.

6.1.2 Model Building:

Our LSTM model was built using Keras's Sequential API. We started with an Embedding layer, which converts our tokenized text data into dense vectors of fixed size. We set the output dimension of our Embedding layer to 128. Next, we added an LSTM layer with 64 units. The LSTM layer analyzes the sequence of word vectors from the Embedding layer and encodes this sequence into a fixed-length vector.

Finally, we added a Dense output layer with a sigmoid activation function. This layer takes the vector from the LSTM layer and outputs a probability that the input text is sarcastic.

6.1.3 Model Training:

We compiled our model with the Adam optimizer and binary cross-entropy loss, which is suitable for our binary classification task. We also decided to monitor accuracy during training. We trained our model for 5 epochs with a batch size of 32. We used 20% of our training data as a validation set to monitor our model's performance on unseen data during training.

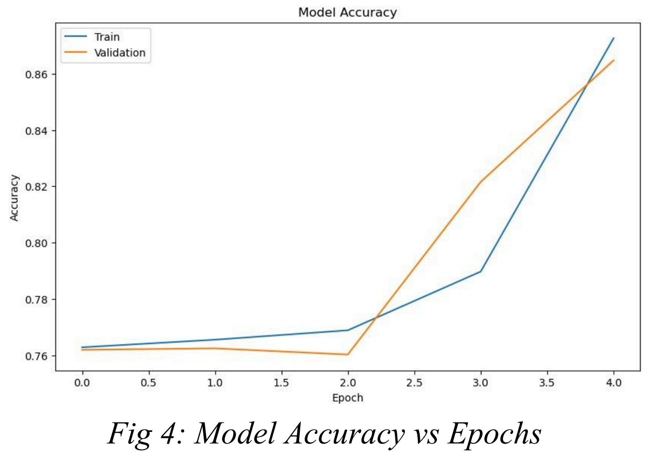


Fig 4: Model Accuracy vs Epochs

The LSTM model achieved an accuracy of 86.87% (Validation accuracy of 85.13%). This suggests that the added complexity of the LSTM model, which can capture longer-term dependencies in the text data, has led to improved performance on this task. LSTM is a powerful model itself.

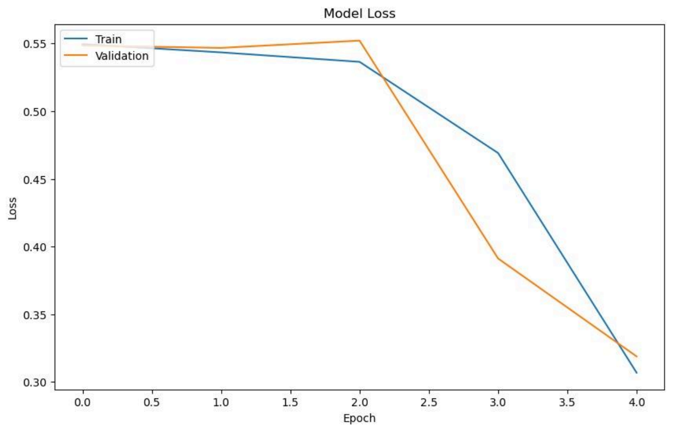


Fig 5: Loss vs Epoch

Fine-tuning the model can potentially lead to improvements in its performance. The sudden increase in accuracy and decrease in loss from epoch 2 to 5 suggests that the model is learning and improving its predictions over time.

6.2 Advanced Modeling

6.2.1 XLNet:

The XLNetTokenizer is used to convert the input text into tokens. These tokens are processed by the XLNetForSequenceClassification model, a variant of the base XLNet model with an additional sequence classification head.

The model and data are moved to a GPU if available; otherwise, they are processed on the CPU. The batch size is set to 8 for both training and datasets. The AdamW optimizer is used for training the model over a specified number of epochs, which is set to 3 in this case. The learning rate for the optimizer is set to 2e-5. In each epoch, the training dataset is iterated over in batches, and the model parameters are updated based on the computed loss. After each epoch, the model's performance is evaluated on the testing dataset, and the accuracy of the model is computed. The training loss was 0.31 and validation accuracy was 87.81%.

6.2.2. BERT:

The implementation utilizes the BERT (Bidirectional Encoder Representations from Transformers) model for a sequence classification task. BERT, a transformer-based model, understands the context of language bidirectionally, generating accurate and contextually rich representations of language.

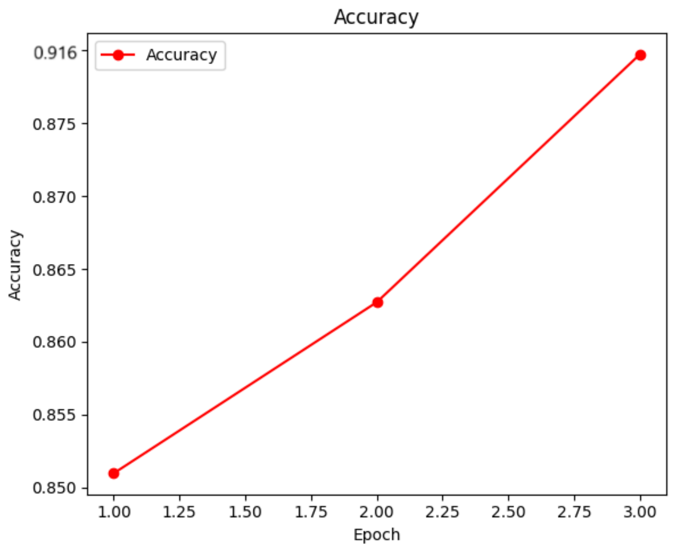


Fig 6: Val Accuracy vs Epochs for BERT

The BertForSequenceClassification model, a variant of the base BERT model with an additional sequence classification head, is used. This head is suitable for tasks such as text classification or sentiment analysis. The model is trained using the AdamW optimizer with a learning rate of 2e-5 over 3 epochs. The batch size for both training and testing datasets is set to 8. The training loss for the mode was 0.09 while accuracy was 91.56%

Fine-tuning pre-trained models like BERT has become a common practice in NLP, often yielding state-of-the-art results on a wide range of tasks. This demonstrates the power and versatility of transformer-based models and their ability to leverage pre-training on large corpora for downstream tasks.

6.2.3 RoBERTa:

RoBERTa model, a variant of BERT that is robustly optimized for text classification tasks. Similar to the previous XLNet example, it involves tokenization of input text, which is then processed by the RobertaForSequenceClassification model. This model is pre-trained and fine-tuned for the task in this code. The training process, including the use of the AdamW optimizer, learning rate, number of epochs, and batch size, is identical to the XLNet example. The model's performance is evaluated on a testing dataset after each epoch, and the accuracy is computed. The training loss and accuracy are 1.8 and 77.93% respectively.

6.2.4 DistilBERT:

A distilled version of BERT that is smaller, faster, cheaper and lighter. The DistilBertTokenizer is used to convert the input text into tokens. These tokens are processed by the DistilBert ForSequenceClassification model, a variant of the base DistilBERT model with an additional sequence classification head. This model is pre-trained and fine-tuned for the specific task in this code. The training process, including the use of the AdamW optimizer, learning rate, number of epochs, and batch size, is identical to the previous examples. The model's performance is evaluated on a testing dataset after each epoch, and the final validation accuracy is computed to be 89.35% while loss was 0.16

Model/ Metric	Accurac y	Precision	Recall	F1 Score	Speed
XLNet	0.88	0.82	0.80	0.80	2
BERT	0.91	0.83	0.72	0.77	3
RoBERT a	0.77	0.71	0.74	0.72	4
DistilBE RT	0.89	0.85	0.68	0.75	1

Table1 : Comparison of the performance of the Transformer Models

BERT turned out to be the best performing model in terms of accuracy while RoBERTa seems unreliable. However DistilBERT is a good choice too since it has a good accuracy and was the fastest model, almost twice as fast as BERT. BERT also, is the most complex model of the list, if enough resources are available, BERT can be used for most accurate results.

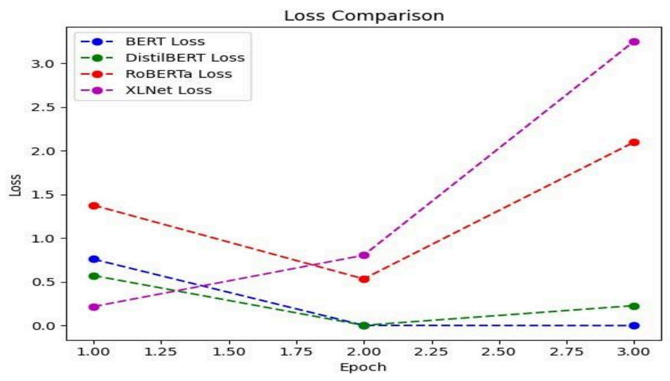


Fig 7: Loss comparison of BERT, DistilBert,RoBERTa and XLNet Models

VII TESTING HYPOTHESIS

After preprocessing the data and conducting the sentiment analysis, a one-way Analysis of Variance (ANOVA) was performed to determine if there were any statistically significant differences between the means of sentiment scores. Following the ANOVA, Tukey’s Honest Significant Difference (HSD) test was performed. This post-hoc test is particularly useful when dealing with data from multiple groups, as it helps control the experiment-wise error rate. Tukey’s HSD(α=0.05) test conducts pairwise comparisons between group means after a one-way ANOVA has been performed. It helped in identifying which specific pairs of subreddit categories have significantly different sentiment scores. The test provided a detailed breakdown of the differences between each pair of communities, allowing for a more granular understanding of the sentiment dynamics within and between various online communities. The results from Tukey’s HSD test were later used to prove the hypothesis, shedding light on the sentiment dynamics within various online communities and contributing to the broader understanding of online discourse.

Algorithm 1 Perform Tukey’s HSD Test

Require: DataFrame df, column names 'sentiment_score' and 'category', significance level alpha

Ensure: posthoc (result of Tukey’s HSD test)

```
1: from statsmodels.stats.multicomp import pairwise_tukeyhsd
2: FUNCTION TukeysHSDTest(df, sentiment_score, category, alpha)
3:     posthoc <- CALL pairwise_tukeyhsd WITH df[sentiment_score],
   df[category], alpha
4:   RETURN posthoc
5: END FUNCTION
6: result <- CALL TukeysHSDTest WITH df, 'sentiment_score',
   'category', 0.05
7: CALL print WITH result
8: END
```

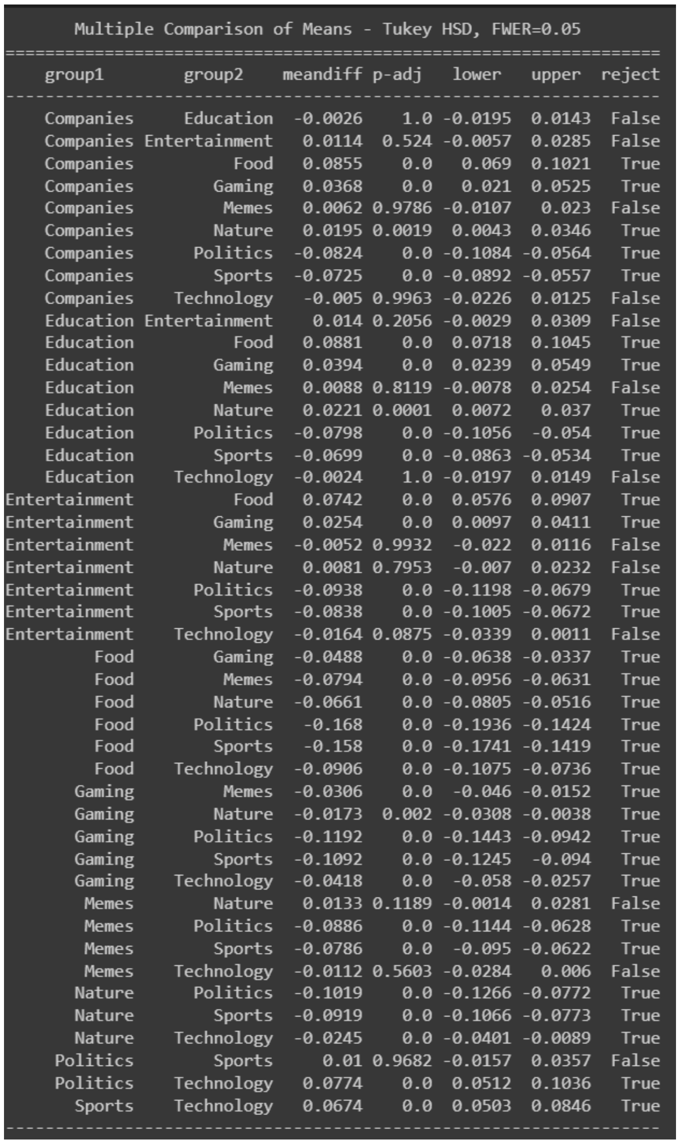
$$Q = \frac{|M_i - M_j|}{\sqrt{MSE \cdot \left(\frac{1}{n_i} + \frac{1}{n_j}\right)}}$$

The formula for Tukey HSD where:
M_i and M_j are the means of the two groups being compared. MSE is the Mean Square Error from the ANOVA table.
n_i and n_j are a number of observations.

VIII RESULTS AND CONCLUSION

The results of the Tukey HSD (Honest Significant Difference) test for multiple comparisons of means revealed significant differences in sentiment across various topics. If ‘True’, it means there is a statistically significant difference in sentiment scores between the two subreddit categories. For instance, the sentiment
scores between the ‘Companies’ and ‘Food’ categories are significantly different, as indicated by the ‘True’ value in the ‘reject’ column as seen in Table 2.

The topics of ‘Food’, ‘Gaming’, and ‘Nature’ elicited significantly more positive sentiment in comparison to ‘Companies’, with mean differences of 0.0855, 0.0368, and 0.0195 respectively, all with a p-value of 0.0. Conversely, the topics of ‘Politics’ and ‘Sports’ attracted significantly more negative sentiment compared to ‘Companies’, with mean differences of -0.0824 and -0.0725 respectively, again with a p-value of 0.0. The distribution of sentiments for ‘Memes’ was pretty even i.e similar number of positive, negative, and neutral comments compared to other subreddits. This is also because of the large number of sarcastic comments in this category. Interestingly, the sentiment towards ‘Education’, ‘Entertainment’, and ‘Technology’ did not significantly differ from that towards ‘Companies’ suggesting that these topics neither attract particularly positive nor negative sentiment.



```
      Multiple Comparison of Means - Tukey HSD, FWER=0.05
==================================================================
   group1        group2    meandiff p-adj   lower   upper  reject
------------------------------------------------------------------
   Companies     Education  -0.0026    1.0 -0.0195  0.0143  False
   Companies Entertainment   0.0114  0.524 -0.0057  0.0285  False
   Companies          Food   0.0855    0.0   0.069  0.1021   True
   Companies        Gaming   0.0368    0.0   0.021  0.0525   True
   Companies         Memes   0.0062 0.9786 -0.0107   0.023  False
   Companies        Nature   0.0195 0.0019  0.0043  0.0346   True
   Companies      Politics  -0.0824    0.0 -0.1084 -0.0564   True
   Companies        Sports  -0.0725    0.0 -0.0892 -0.0557   True
   Companies    Technology   -0.005 0.9963 -0.0226  0.0125  False
   Education Entertainment    0.014 0.2056 -0.0029  0.0309  False
   Education          Food   0.0881    0.0  0.0718  0.1045   True
   Education        Gaming   0.0394    0.0  0.0239  0.0549   True
   Education         Memes   0.0088 0.8119 -0.0078  0.0254  False
   Education        Nature   0.0221 0.0001  0.0072   0.037   True
   Education      Politics  -0.0798    0.0 -0.1056  -0.054   True
   Education        Sports  -0.0699    0.0 -0.0863 -0.0534   True
   Education    Technology  -0.0024    1.0 -0.0197  0.0149  False
Entertainment         Food   0.0742    0.0  0.0576  0.0907   True
Entertainment       Gaming   0.0254    0.0  0.0097  0.0411   True
Entertainment        Memes  -0.0052 0.9932  -0.022  0.0116  False
Entertainment       Nature   0.0081 0.7953  -0.007  0.0232  False
Entertainment     Politics  -0.0938    0.0 -0.1198 -0.0679   True
Entertainment       Sports  -0.0838    0.0 -0.1005 -0.0672   True
Entertainment   Technology  -0.0164 0.0875 -0.0339  0.0011  False
         Food       Gaming  -0.0488    0.0 -0.0638 -0.0337   True
         Food        Memes  -0.0794    0.0 -0.0956 -0.0631   True
         Food       Nature  -0.0661    0.0 -0.0805 -0.0516   True
         Food     Politics   -0.168    0.0 -0.1936 -0.1424   True
         Food       Sports   -0.158    0.0 -0.1741 -0.1419   True
         Food   Technology  -0.0906    0.0 -0.1075 -0.0736   True
       Gaming        Memes  -0.0306    0.0  -0.046 -0.0152   True
       Gaming       Nature  -0.0173  0.002 -0.0308 -0.0038   True
       Gaming     Politics  -0.1192    0.0 -0.1443 -0.0942   True
       Gaming       Sports  -0.1092    0.0 -0.1245  -0.094   True
       Gaming   Technology  -0.0418    0.0  -0.058 -0.0257   True
        Memes       Nature   0.0133 0.1189 -0.0014  0.0281  False
        Memes     Politics  -0.0886    0.0 -0.1144 -0.0628   True
        Memes       Sports  -0.0786    0.0  -0.095 -0.0622   True
        Memes   Technology  -0.0112 0.5603 -0.0284   0.006  False
       Nature     Politics  -0.1019    0.0 -0.1266 -0.0772   True
       Nature       Sports  -0.0919    0.0 -0.1066 -0.0773   True
       Nature   Technology  -0.0245    0.0 -0.0401 -0.0089   True
     Politics       Sports     0.01 0.9682 -0.0157  0.0357  False
     Politics   Technology   0.0774    0.0  0.0512  0.1036   True
       Sports   Technology   0.0674    0.0  0.0503  0.0846   True
------------------------------------------------------------------
```

Table 2: Tukey HSD Test Resulting Table

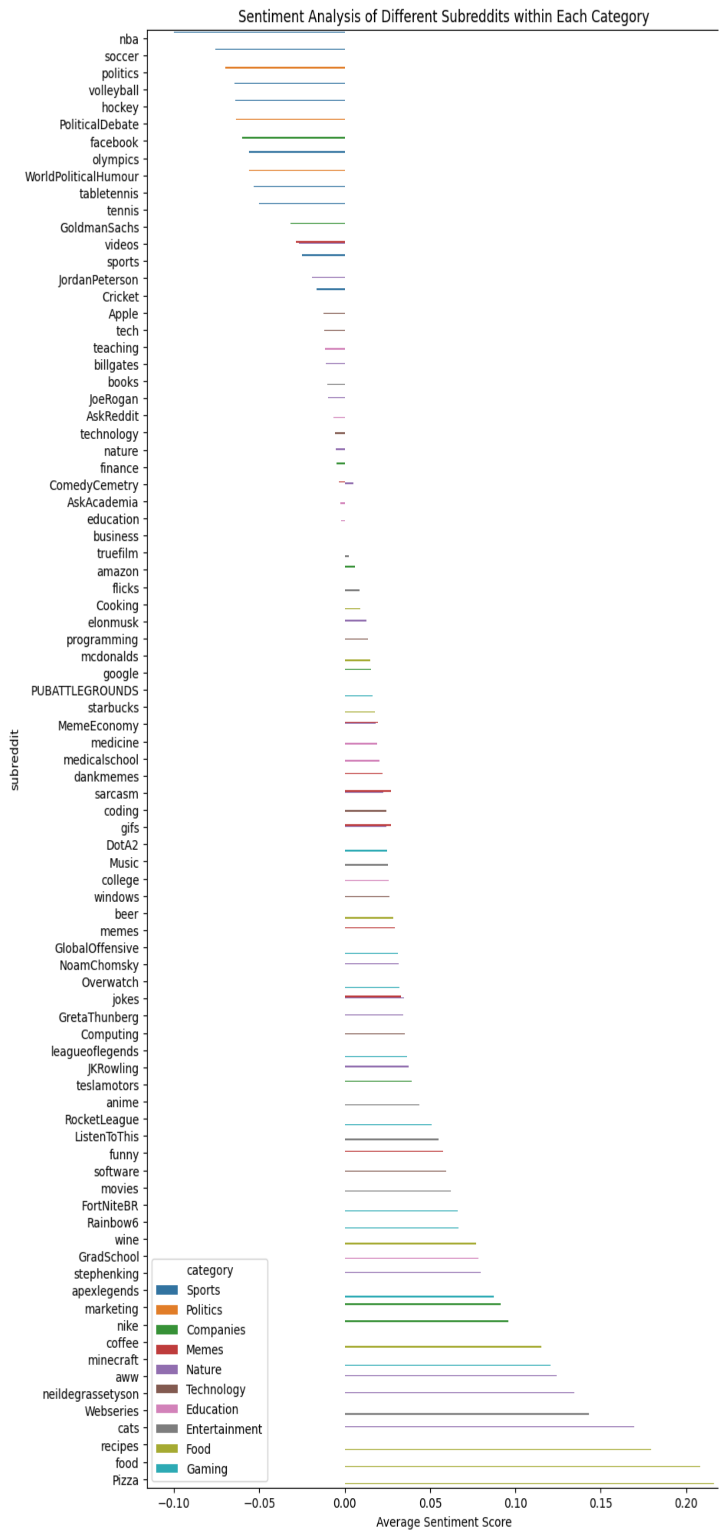


Fig 8: Comparison of sentiment distribution across subreddits of different categories

Fig 8 Analyzes the sentiment of comments in different subreddits within each category, and visualizes the results with a bar chart. These results are consistent with the statistical testing results as the subreddits with higher average sentiment scores 'Pizza', 'food' belong to the same category and subreddits related to Politics and Sports are much lower. There are some interesting deviations though. Facebook and Goldman Sachs have surprisingly lower sentiment scores than few in the Sports category, while others in the same category like Tesla and Amazon are on the higher sides. Thus, Even amongst comments in the same category, some subreddits tend to vary in the range of positive and negative sentiments. Some other eye-catching results are Webseries beating Books, but this can be attributed to the fact that most comments in r/books were neutral while webseries have a huge fan base these days and fans tend to flood in the comments with positives of their favorite shows and this overshadows the critics.

Category	**Subreddit**	**Comment Body**	**Sentiment Score**	**Sarcastic Probability**	**Label**
Tech	*r/programming*	Can I use processing (p5.js)? Wanted to know more but the site is down (	0.0000	0.21156	Neutral
Politics	*r/politics*	Send him to The Hague	-0.3151	0.5614	Negative
Food	*r/Pizza*	Love his expression at the end. "No biggie"	0.5601	0.3204	Positive

Table 3: Random Comments from the dataset labeled

Neutral comments are higher in numbers in categories like Technology and Companies. A reason for this can be attributed to most of the comments in such topics are facts and not opinions of people. Sentiments lie on the extremes(Positive and negative) more often in topics that attract people to strongly put their opinion like sports, webseries, politics, etc. An example is given in Table 3.

In conclusion, our hypothesis was confirmed. Certain topics on Reddit do indeed attract more positive or negative sentiment, but this is not universal across all topics. These findings provide valuable insights into the sentiment dynamics within different subreddits and can inform community management strategies and content creation guidelines. Additionally, for text classification tasks like Sentiment Analysis, pretrained transformers can be a go-to option. Future research could explore the reasons behind these sentiment differences and investigate how they evolve over time. Other social media platforms like Twitter allow a spectrum of visual data in the comments. Neural networks would be needed to deal with this in the preprocessing stage itself.

References:

[1] Abiona A.A. et al, International Journal of Computer Science and Mobile Computing, Vol.11 Issue.9, September- 2022, pg. 34-40
[2] H. T. Phan, V. C. Tran, N. T. Nguyen and D. Hwang, "Improving the Performance of Sentiment Analysis of Tweets Containing Fuzzy Sentiment Using the Feature Ensemble Model," in IEEE Access, vol. 8, pp.14630-14641, 2020, doi: 10.1109/ACCESS.2019.2963702
[3] Y. Wang, G. Huang, J. Li, H. Li, Y. Zhou and H. Jiang, "Refined Global Word Embeddings Based onSentiment Concept for Sentiment Analysis," in IEEE Access, vol. 9, pp. 37075-37085, 2021, doi:10.1109/ACCESS.2021.3062654
[4] Guerra A, Karakuş O. Sentiment analysis for measuring hope and fear from Reddit posts during the 2022 Russo-Ukrainian conflict. Front Artif Intell. 2023 Apr 5;6:1163577. doi: 10.3389/frai.2023.1163577. PMID: 37091300; PMCID: PMC10113549.
[5] Wankhade, M., Rao, A.C.S. & Kulkarni, C. A survey on sentiment analysis methods, applications, and challenges. *Artif Intell Rev* 55, 5731–5780 (2022). https://doi.org/10.1007/s10462-022-10144-1.
[6] M. Völske, M. Potthast, S. Syed, and B. Stein, "TL;DR: Mining Reddit to Learn Automatic Summarization," in Proceedings of the Workshop on New Frontiers in Summarization, Copenhagen, Denmark, Sep. 2017, pp. 59-63, doi: 10.18653/v1/W17-4508
[7]Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, Omer Levy, “LIMA: Less Is More for Alignment”, arXiv:2305.11206v1 [cs.CL] 18 May 2023.
[8]Mikhail Khodak and Nikunj Saunshi and Kiran Vodrahalli, “A Large Self-Annotated Corpus for Sarcasm”, https://arxiv.org/abs/1704.05579, year=2017
[9]Lindskog, Sebastian and Serur, Juan Andrés, Reddit Sentiment Analysis (August 15, 2020).

Links:
https://en.wikipedia.org/wiki/Tukey%27s_range_test