Detecting Political Sarcasm: A Comparative Study of Traditional NLP and Transformer Models

Henry Vu
ddv240000@utdallas.edu

May 20, 2025

Abstract

Sarcasm detection, particularly in nuanced domains like political discourse, remains a significant challenge in Natural Language Processing (NLP). This work presents a comprehensive investigation into sarcasm detection using a large dataset of Reddit comments marked with '/s'. I focus specifically on comments from politically oriented subreddits, reflecting a domain where sarcasm is prevalent and often subtle. My methodology encompasses both traditional NLP techniques and modern transformer-based models. For traditional methods, I implement Bag-of-Words (BoW), TF-IDF, and TF-IDF with N-grams, significantly enhanced by a novel suite of engineered features. I evaluate Logistic Regression, Random Forest, and ensemble models on these features. Complementing this, I fine-tune DistilRoBERTa, a transformer model, analyzing its performance and interpretability through attention mechanisms and token importance. Furthermore, I employ Latent Dirichlet Allocation (LDA) topic modeling on sarcastic and non-sarcastic comments to uncover underlying thematic differences. My results demonstrate the superior performance of the fine-tuned DistilRoBERTa model, surpassing the best traditional model. This study provides a detailed comparison of methodologies, highlights the complexity of political sarcasm detection, and offers insights through extensive visualization and analysis.

1 Introduction

The proliferation of user-generated content on social media platforms like Reddit has created vast repositories of text data, offering invaluable insights into public opinion, social dynamics, and linguistic evolution. However, interpreting this data accurately is often confounded by figurative language, among which sarcasm stands out as particularly challenging. Sarcasm, typically involving the use of words that mean the opposite of their literal intention, relies heavily on context, tone (absent in text), and shared knowledge, making its automatic detection a complex NLP task.

Political discourse is a domain rife with sarcasm. It serves various functions, from expressing dissent and criticism humorously to reinforcing group identity. Misinterpreting sarcasm in this context can lead to misunderstandings, polarization, and flawed analyses of political sentiment. Therefore, I consider developing robust sarcasm detection models important.

This research focuses on detecting sarcasm within political comments sourced from Reddit. I leverage a large dataset where users explicitly self-annotate sarcasm using the '/s' marker, providing a valuable, albeit potentially biased, ground truth. My primary contributions are:

1) **Focused Data Curation:** I isolate comments from a comprehensive list of politically relevant subreddits, creating a domain-specific dataset for analysis.
2) **Novel Feature Engineering:** I design and implement an extensive set of features for traditional NLP models, going beyond standard text representations to include linguistic cues (punctuation, capitalization), sentiment dynamics (polarity, subjectivity, contrast across sentences), readability metrics, sarcasm-specific markers (intensifiers, contrast words, slang), and contextual information derived from subreddit statistics.
3) **Comparative Model Evaluation:** I systematically evaluate classic NLP approaches (BoW, TF-IDF, N-grams with Logistic Regression, Random Forest, Ensembles) against a fine-tuned transformer model (DistilRoBERTa), providing a clear performance comparison on the curated political dataset.
4) **In-depth Analysis and Interpretability:** I employ various techniques to understand model behavior, including feature importance analysis for traditional models, attention mechanism visualization and token importance analysis for the transformer model, and detailed error analysis focusing on misclassifications and model disagreements.
5) **Thematic Exploration via LDA:** I experiment with Latent Dirichlet Allocation to model the underlying topics within sarcastic and non-sarcastic comments separately, which helps identify key thematic differences in politically sarcastic comments on Reddit.
6) **Extensive Visualization:** I incorporate a wide array of visualizations to aid understanding, including word clouds, performance comparison charts, confusion matrices, ROC/PR curves, training progress plots, attention heatmaps, token importance plots, and LDA topic comparison diagrams.

This report details my methodology, experimental setup, results, and analysis, culminating in a discussion of lessons learned and potential avenues for future research in political sarcasm detection. For any specific information regarding implementation, please refer to the notebooks on GitHub (SarcasmNLP.ipynb and SarcasmLLM.ipynb)

Project Link: GitHub

2 Methodology

My approach integrates data preprocessing, traditional NLP pipeline development with advanced feature engineering, transformer model fine-tuning, topic modeling, and rigorous evaluation.

2.1 Data Acquisition and Preprocessing

The primary data source is the publicly available Reddit Sarcasm Dataset on Kaggle, which contains comments labeled as sarcastic (1) or non-sarcastic (0), often based on the presence of the '/s'marker.

Domain Filtering: Recognizing the unique characteristics of political language, I filtered the dataset to retain comments exclusively from a predefined list of 31 politically-oriented subreddits (such as r/politics, r/worldnews, r/The_Donald, r/Conservative, r/progressive, etc.). This step ensures the relevance of my models to the target domain.

Data Cleaning: Initial cleaning involved removing rows with missing values in essential columns (comment, label, subreddit).

Text Preprocessing Pipeline: For both traditional NLP and LDA, I applied the following text preprocessing steps to the raw 'comment' field:

- Convert text to lowercase.
- Remove special characters (non-alphanumeric, excluding whitespace, digits) using regular expressions.
- Tokenize the text into words using NLTK's 'word_tokenize'.
- Remove standard English stopwords using 'nltk.corpus.stopwords'.
- Filter out tokens with length less than or equal to 2.
- Rejoin tokens into a processed string.

Comments that became empty after preprocessing were removed. The transformer model utilized its own tokenizer's preprocessing capabilities.

Data Splitting: The filtered and cleaned political comment data was split into training (64% or 70486 comments), validation (16% or 17622 comments), and test (20% or 22028 comments) sets using stratified sampling based on the 'label'to maintain class distribution across splits. The random seed was fixed at 42 for reproducibility.

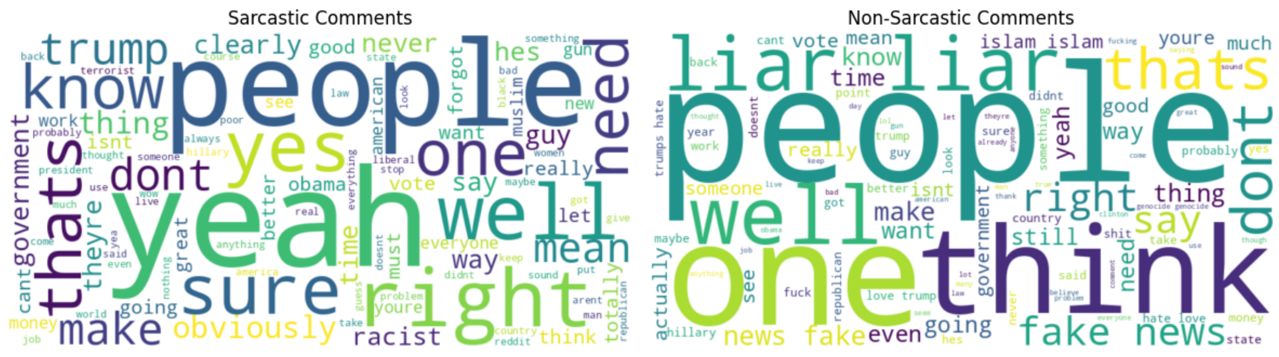


Figure 1: Word cloud of processed Sarcastic vs Non-Sarcastic comments.

2.2 Traditional NLP Approaches (SarcasmNLP.ipynb)

I explored several traditional NLP techniques, focusing on feature engineering to capture sarcasm cues.

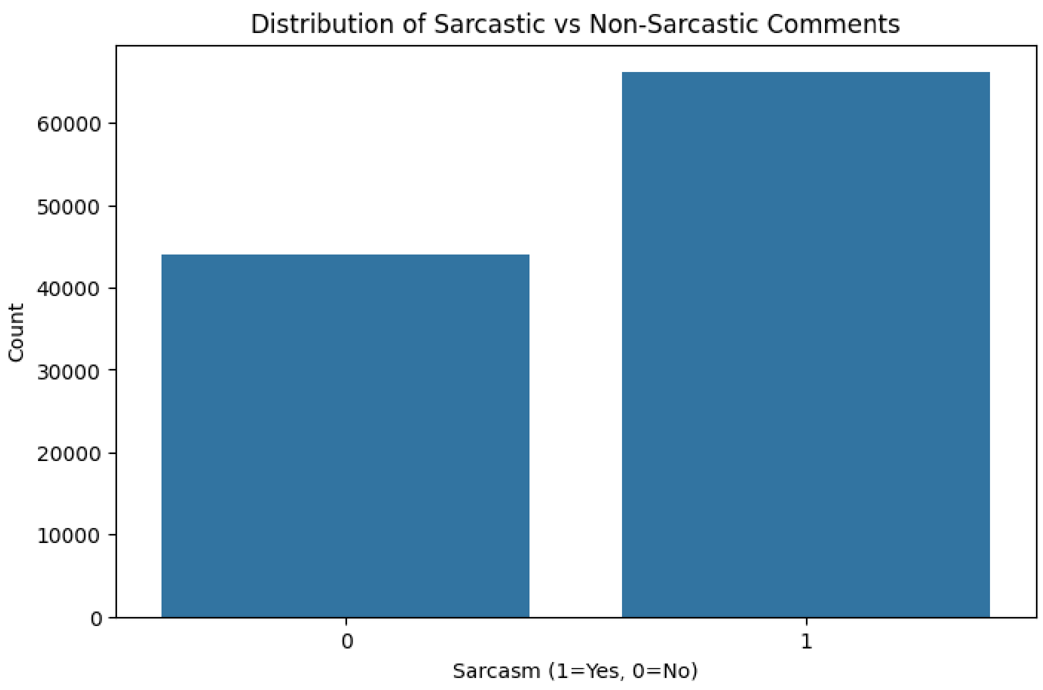


Figure 2: Distribution of Sarcastic vs Non-Sarcastic comments.

2.2.1 Feature Engineering (Novelty Aspect)

Beyond standard text vectorization, I engineered a rich set of features, categorized as follows:

- **Linguistic Features:** Capturing stylistic elements often associated with expressive or sarcastic language.
  - Punctuation Counts: Number of exclamation marks (!), question marks (?), quotation marks (”, ’), ellipses (...).
  - Capitalization Ratio: Proportion of fully capitalized words (length > 1) to total words.
  - Word Count: Total number of words.
  - Average Word Length: Mean length of words in the comment.
  - Unique Words Ratio: Ratio of unique words to total words (lexical diversity).
- **Sarcasm-Specific Markers:** Targeting features identified in prior literature as potential sarcasm indicators.
  - Intensifier Count: Presence of words like ‘really’, ‘very’, ‘so‘, ‘totally‘, ‘absolutely‘.
  - Contrast Word Count: Presence of words like ‘but‘, ‘however‘, ‘although‘.
  - Internet Slang Count: Presence of terms like ‘lol‘, ‘lmao‘, ‘rofl‘.
  - Exaggeration Markers: Detection of patterns like repeated letters (e.g., ‘sooo‘), specific spellings (‘riiight‘), or phrases (‘definitely not‘).
  - Quotation Mark Usage: Count of quotation marks, potentially used for emphasis or distancing.

    - Emoticon Presence: Binary flag indicating the presence of common emoticons.

- **Sentiment Analysis Features (TextBlob):** Measuring emotional tone and its dynamics.
    - Overall Polarity: Sentiment score from -1 (negative) to 1 (positive).
    - Overall Subjectivity: Score from 0 (objective) to 1 (subjective).
    - Sentiment Contrast: Binary flag indicating if the comment contains both significantly positive (>0.2) and significantly negative (<-0.2) sentences.
    - Sentiment Standard Deviation: Standard deviation of polarity scores across sentences, capturing sentiment volatility.
- **Contextual Features:** Leveraging metadata associated with the comment.
    - Subreddit Sarcasm Rate: The historical average sarcasm rate (label mean) of the subreddit the comment belongs to, calculated on the training set.
- **Readability and Complexity Metrics:** Assessing the linguistic complexity.
    - Average Sentence Length: Mean number of words per sentence.
    - Readability Score: Approximation of Flesch Reading Ease score, normalized to 0-1.

These features were calculated for each comment and combined with text vectorizations.

2.2.2 Vectorization Techniques

I employed standard text vectorization methods on the 'processed_comment':

- **Bag-of-Words (BoW):** Using 'CountVectorizer' with 'max_features=5000'.
- **Term Frequency-Inverse Document Frequency (TF-IDF):** Using 'TfidfVectorizer' with 'max_features=5000'.
- **TF-IDF with N-grams:** Using 'TfidfVectorizer' with 'max_features=7000' and 'ngram_range=(1, 3)' to capture unigrams, bigrams, and trigrams.

2.2.3 Classification Models

The following classifiers were trained and evaluated:

- **Logistic Regression:** A standard linear baseline ($C = 1.0$, max_iter = 1000).
- **Hyperparameter Tuned Logistic Regression:** Using GridSearchCV (cv=3, scoring='f1_weighted') and the TF-IDF N-gram features to find optimal C, class_weight, and solver.
- **Random Forest:** An ensemble tree-based model (n_estimators=100).

- **Voting Classifier (Ensemble):** A soft voting ensemble combining predictions from the BoW Logistic Regression, TF-IDF Logistic Regression, and the Tuned TF-IDF N-gram Logistic Regression. (Note: Due to memory constraints, this was potentially trained/evaluated on dense representations or subsets).

Models were trained both *with* and *without* the engineered features concatenated (using 'scipy.sparse.hstack' for sparse matrices) to assess the feature impact.

2.3 Transformer-Based Approach (SarcasmLLM)

I utilized a pre-trained transformer model for sequence classification.

2.3.1 Model Selection

I selected **DistilRoBERTa-base** ('distilroberta-base'). This choice represents a balance between the strong performance of RoBERTa-style models and computational efficiency, making fine-tuning feasible. RoBERTa's training objective (dynamic masking, no NSP) is often beneficial for downstream tasks compared to BERT. The model was configured for binary classification ('num_labels=2') and to output attentions and hidden states for analysis.

2.3.2 Tokenization

I used the 'RobertaTokenizer' corresponding to 'distilroberta-base'. Input comments were tokenized using 'encode_plus' with the following settings:

- 'add_special_tokens=True': Adds '$< s >$' (CLS) and '<\s>' (SEP) tokens.
- 'max_length=128': Truncates or pads sequences to a maximum length of 128 tokens.
- 'padding='max_length'': Pads shorter sequences to 'max_length'.
- 'truncation=True': Truncates longer sequences.
- 'return_attention_mask=True': Generates an attention mask to ignore padding tokens.
- 'return_tensors='pt'': Returns PyTorch tensors.

2.3.3 Dataset and Fine-tuning

A custom PyTorch 'Dataset' class ('SarcasmDataset') was created to handle tokenization on-the-fly. DataLoaders were used for efficient batching ('batch_size=16') and shuffling during training.

The model was fine-tuned using the following setup:

- **Optimizer:** AdamW (‘AdamW(model.parameters(), lr=2e-5)‘).
- **Scheduler:** Linear schedule with no warmup (‘get_linear_schedule_with_warmup‘ with ‘num_warmup_steps=0‘). The total number of training steps was calculated based on the training data size and number of epochs.
- **Training Loop:** Standard PyTorch training loop involving forward pass, loss calculation (cross-entropy loss provided by the model’s output), backward pass, optimizer step, and scheduler step.
- **Epochs:** The model was trained for ‘epochs=3‘.
- **Device:** Training was performed on a GPU (‘cuda‘) if available, otherwise CPU.
- **Best Model Saving:** The model weights and tokenizer configuration yielding the best F1-score on the validation set during training were saved.

2.4 Latent Dirichlet Allocation (LDA) Topic Modeling

To explore the thematic content associated with sarcasm, I applied LDA separately to the processed sarcastic and non-sarcastic comments from the training set.

- **Vectorization:** ‘CountVectorizer‘ was used (‘max_features=5000‘, ‘min_df=5‘, ‘max_df=0.9‘, ‘stop_words=’english’‘).
- **LDA Model:** ‘LatentDirichletAllocation‘ from scikit-learn was used with ‘n_components=5‘ (5 topics), ‘max_iter=50‘, ‘learning_method=’online’‘, and adjusted priors (‘doc_topic_prior=0.5‘, ‘topic_word_prior=0.05‘) based on observed notebook parameters. The same vectorizer vocabulary was used for both sarcastic and non-sarcastic LDA models.
- **Analysis:** I extracted the top words for each topic, calculated document-topic distributions, identified dominant topics per document, and compared topic prevalence between sarcastic and non-sarcastic sets.

2.5 Evaluation Metrics

Model performance was evaluated using standard classification metrics:

- Accuracy
- Precision (per class and weighted average)
- Recall (per class and weighted average)
- F1-Score (per class and weighted average)
- Confusion Matrix
- Area Under the Receiver Operating Characteristic Curve (ROC AUC)

- Area Under the Precision-Recall Curve (PR AUC)

Training time was also recorded for the transformer model.

3 Experiments and Results

I present the performance of the different modeling approaches on the held-out validation and test sets derived from the political Reddit comments. I follow the typical fashion of NLP research and report the weighted average of accuracy and F-1 Score.

3.1 Traditional NLP Model Performance

The performance of traditional NLP models, evaluated on the validation set, is summarized below. I primarily focus on the results obtained when combining text vectorization with the engineered features, as this generally yielded better performance. The Tuned TF-IDF N-gram model showed the strongest performance among traditional methods.

Table 1: Performance of Traditional NLP Models on the Validation Set (without Engineered Features).

Model	Accuracy	F1 Score (Weighted)
BoW + Features	0.6915	0.6880
TF-IDF + Features	0.6949	0.6899
TF-IDF N-gram + Features	0.6949	0.6899
Tuned TF-IDF N-gram + Features	**0.6953**	**0.6904**
Random Forest (TF-IDF N-gram Feat)	0.6944	0.6900
Ensemble (Soft Voting) + Features	0.6983	0.6951

Table 2: Performance of Traditional NLP Models on the Validation Set (with Engineered Features).

Model	Accuracy	F1 Score (Weighted)
BoW + Features	0.7116	0.7094
TF-IDF + Features	0.7123	0.7090
TF-IDF N-gram + Features	0.7123	0.7090
Tuned TF-IDF N-gram + Features	**0.7126**	0.7092
Random Forest (TF-IDF N-gram Feat)	0.7053	0.6990
Ensemble (Soft Voting) + Features	0.7123	**0.7099**

The inclusion of engineered features consistently provided a performance boost compared to using text vectorization alone (detailed comparison omitted for brevity, but observed during development). Figure 3 visualizes the accuracy and F1-scores. Confusion matrices (Figure 4) illustrate the true positive/negative and false positive/negative rates for key models.

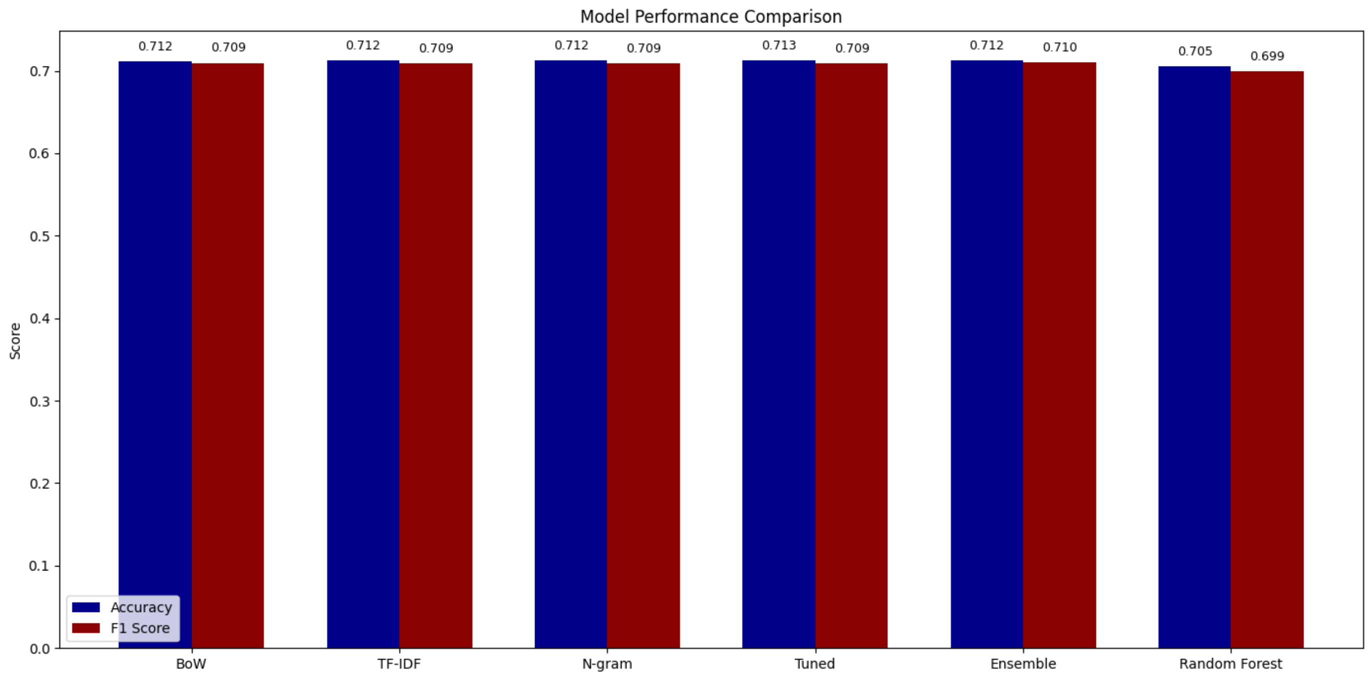


Figure 3: Performance Comparison of Traditional NLP Models (Accuracy and F1 Score) on the Validation Set.

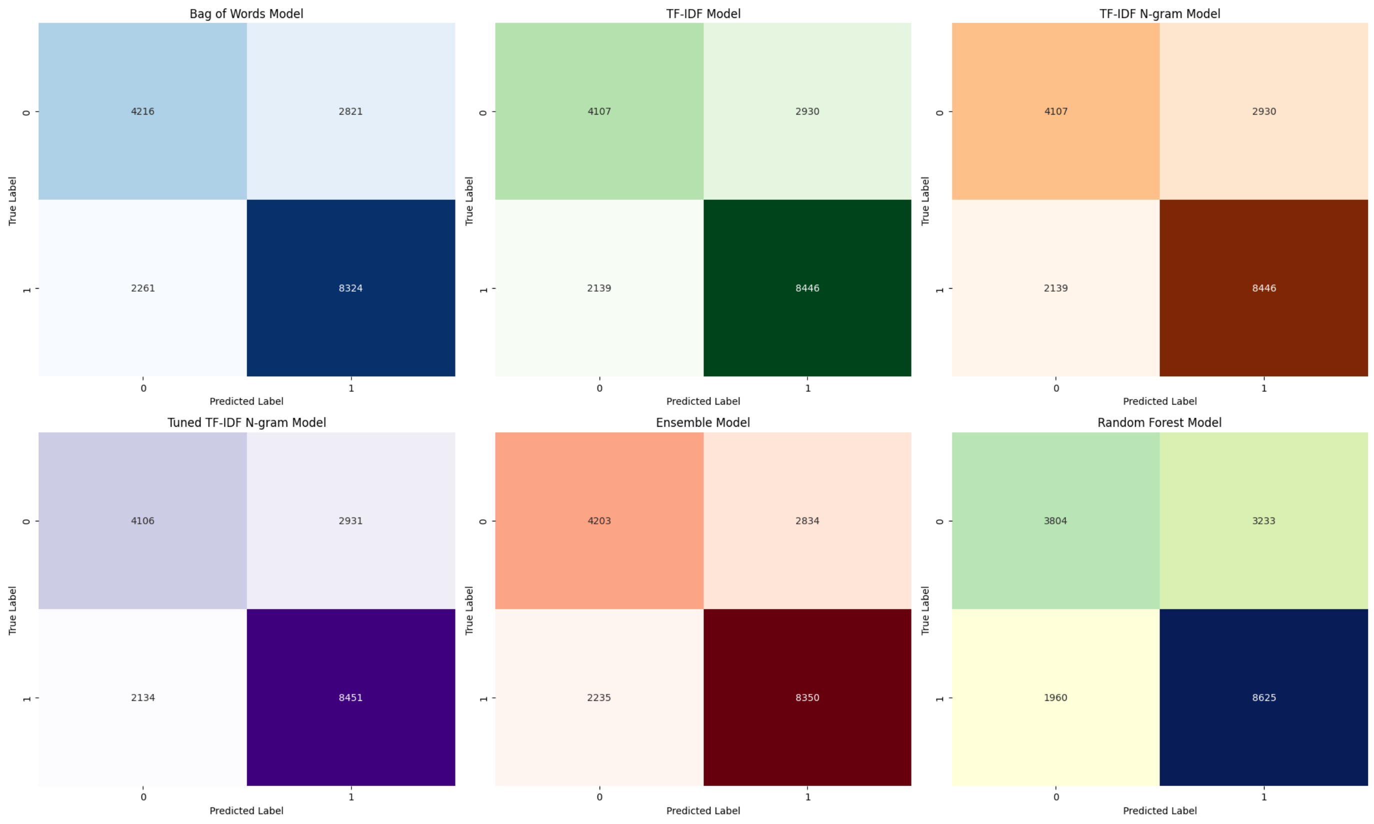


Figure 4: Confusion Matrix of different models.

3.2 Transformer Model Performance

I evaluated the base DistilRoBERTa model from HuggingFace prior to further fine-tuning. The base model achieved a weighted average F1 score of 0.23, and accuracy of 0.4019. This is a worse result than a random guess.

To improve the performance, the DistilRoBERTa model was fine-tuned for 5 epochs. The training progress, showing loss and accuracy on both training and validation sets per epoch, is depicted in Figure 5.

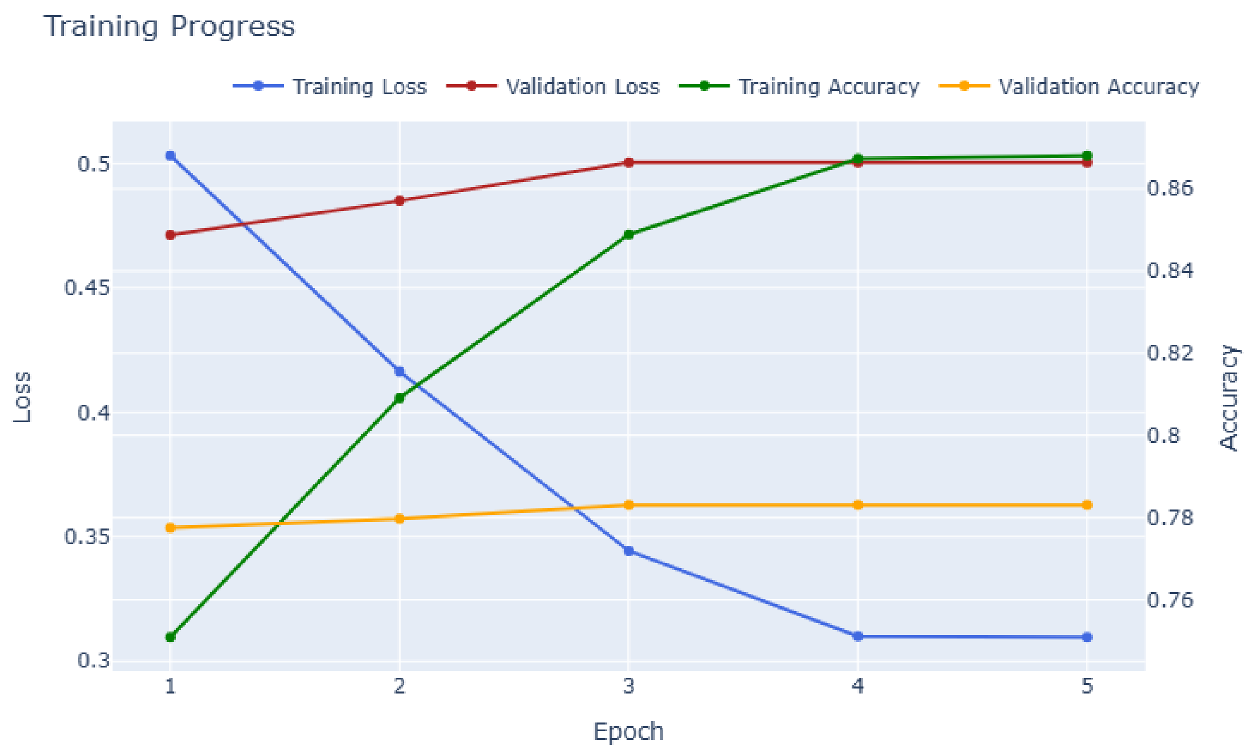


Figure 5: DistilRoBERTa Fine-tuning Progress: Training and Validation Loss and Accuracy over Epochs.

The best model checkpoint (based on validation F1-score) was evaluated on the held-out test set. The performance metrics are presented in Table 3 and compared against baseline models in Figure 8. The total training time was approximately 2356.93 seconds on the available hardware (L4 GPU).

Table 3: Performance of Fine-tuned DistilRoBERTa on the Test Set.

Metric	Value
Accuracy	0.7833
F1 Score (Weighted)	0.7814
ROC AUC	0.864
PR AUC	0.903

Evaluation visualizations for the transformer model on the test set are provided: Confusion Matrix (Figure 6), ROC Curve (Figure 7a), and Precision-Recall Curve (Figure 7b).

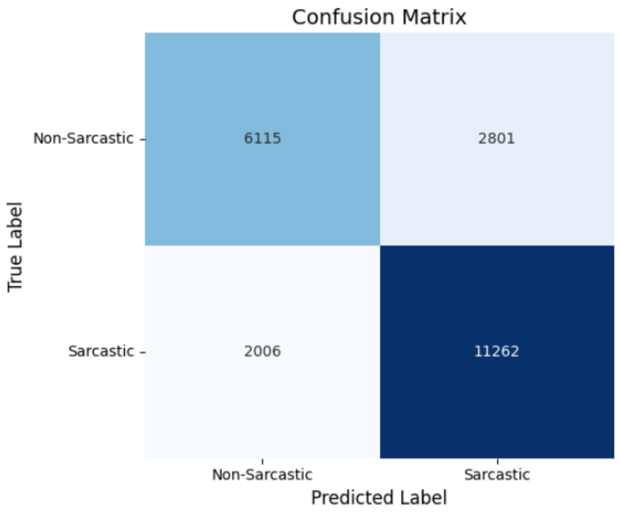
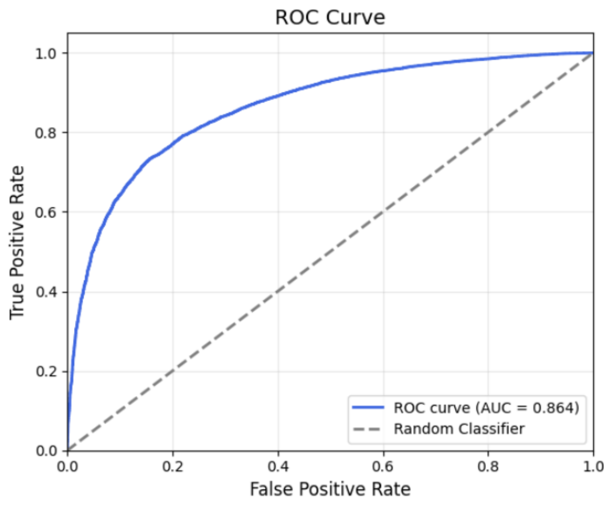
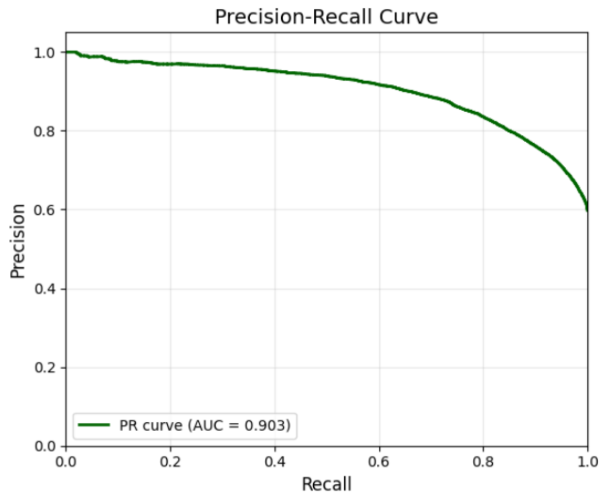


Figure 6: Confusion Matrix for DistilRoBERTa on the Test Set.



(a) ROC Curve (AUC = 0.864).



(b) Precision-Recall Curve (AUC = 0.903).

Figure 7: ROC and Precision-Recall Curves for DistilRoBERTa on the Test Set.

Figure 8 provides a direct comparison between the fine-tuned DistilRoBERTa and the baseline traditional models evaluated in the ‘SarcasmLLM‘ notebook (BoW, TF-IDF).

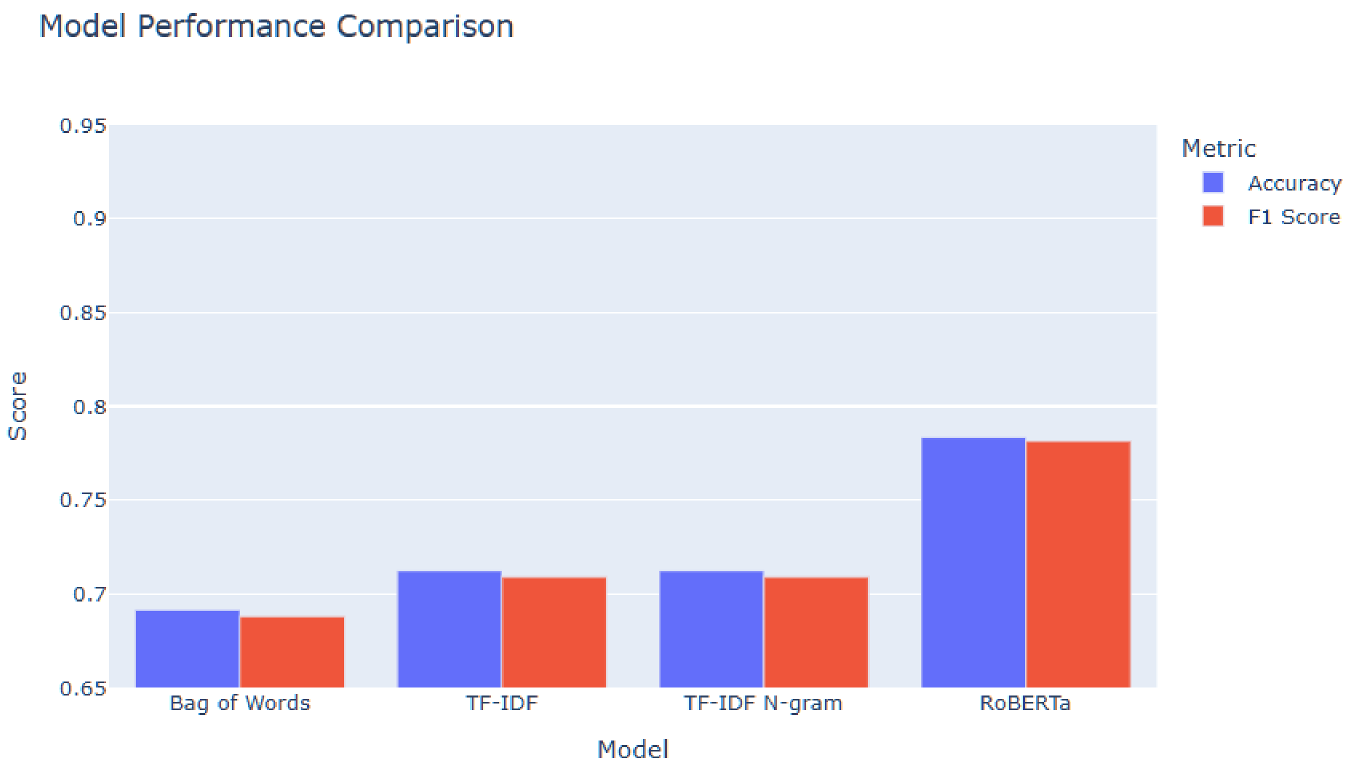


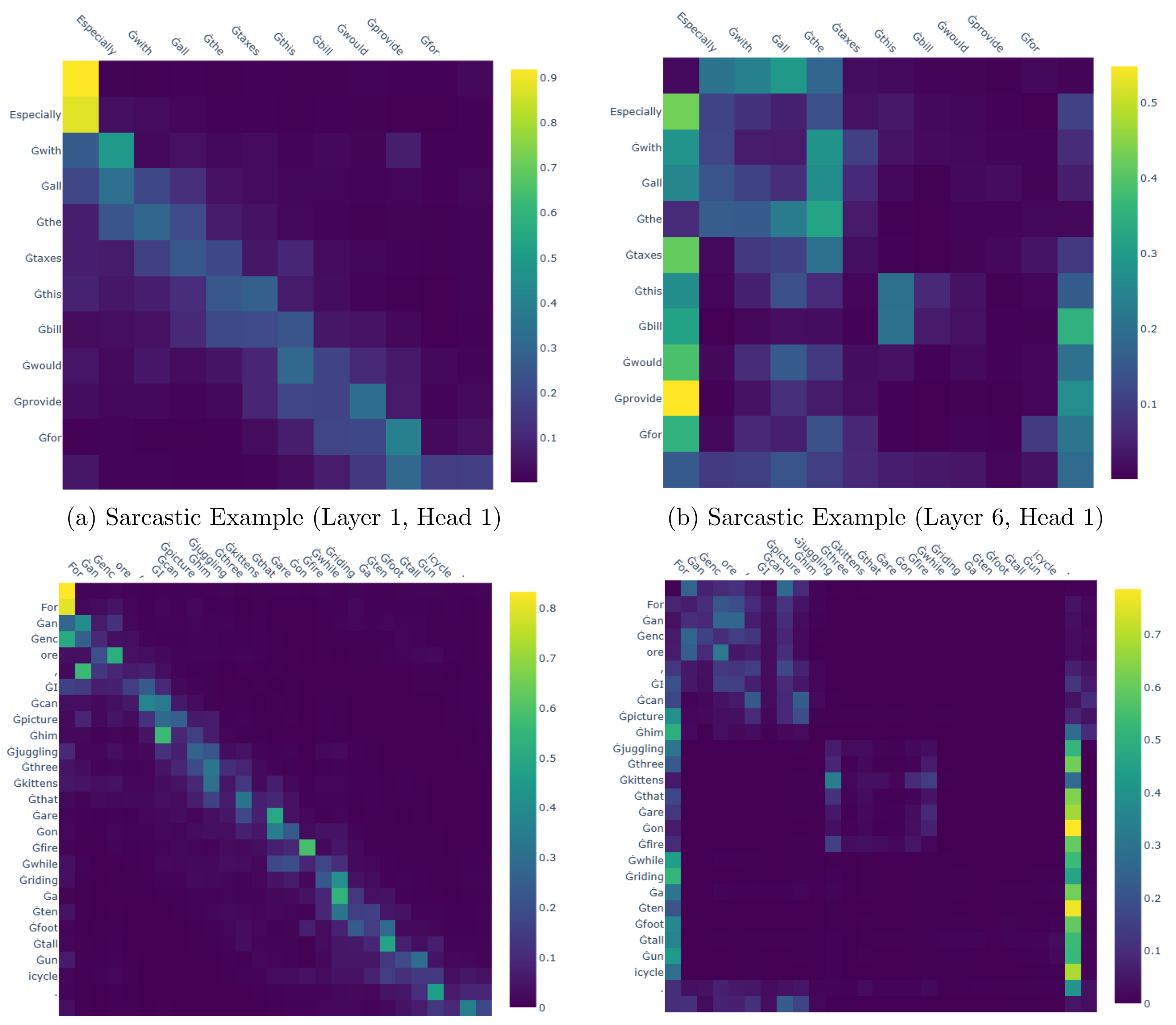
Figure 8: Performance Comparison: DistilRoBERTa vs. Baseline Traditional Models.

3.3 Model Interpretability and Analysis

I conducted several analyses to understand model behavior and identify challenging aspects of the task.

3.3.1 Attention Mechanism Visualization (Transformer)

To gain insight into how DistilRoBERTa processes input sequences, I visualized the self-attention weights from different layers and heads using attention heatmaps (Figure 9). I examined attention patterns for selected sarcastic and non-sarcastic examples across different layers (e.g., early, middle, late - layers 0, 2, 5 in DistilRoBERTa).



(a) Sarcastic Example (Layer 1, Head 1)

(b) Sarcastic Example (Layer 6, Head 1)

(c) Non-Sarcastic Example (Layer 1, Head 1)

(d) Non-Sarcastic Example (Layer 6, Head 1)

Figure 9: Attention Head Visualization Examples for Sarcastic and Non-Sarcastic Comments across Different Layers.

We can observe that at the beginning, neighboring tokens unsurprisingly have higher attention score compared to tokens that are far apart. However, it is hard to describe any contrasting patterns between Sarcastic and Non-Sarcastic comments in general.

3.3.2 Error Analysis

I investigated misclassified examples from the transformer model's predictions on the test set.

- **High-Confidence Errors:** I examined instances where the model made an incorrect prediction with high confidence (softmax probability $>$ [e.g., 0.8]). Examples:

  - Example 1 from r/politics: ”I am republican-leaning mostly, but I still don’t think that the Democratic Party is off by as far as this title suggests...” True: Sarcastic, Predicted: Non-Sarcastic, Confidence: 0.9964
  - Example 2 from r/politics: ”There’s a large group of actions by this non acting administration supporting your bullshit argument...” True: Sarcastic, Predicted: Non-Sarcastic, Confidence: 0.9949

  These examples highlight several challenges. The first case from `r/politics` demonstrates subtle sarcasm; the phrasing appears as measured disagreement, which the model likely interpreted literally, missing the potentially sarcastic exaggeration implied about the ”title”. The second `r/politics` example is intriguing; despite containing strong negative language (”non acting administration”, ”bullshit argument”), the model failed to detect the sarcasm inherent in linking these actions to supporting the argument, perhaps focusing too much on the literal structure.

- **Subreddit Error Rates:** I analyzed the misclassification rate across different subreddits (Figure 10).

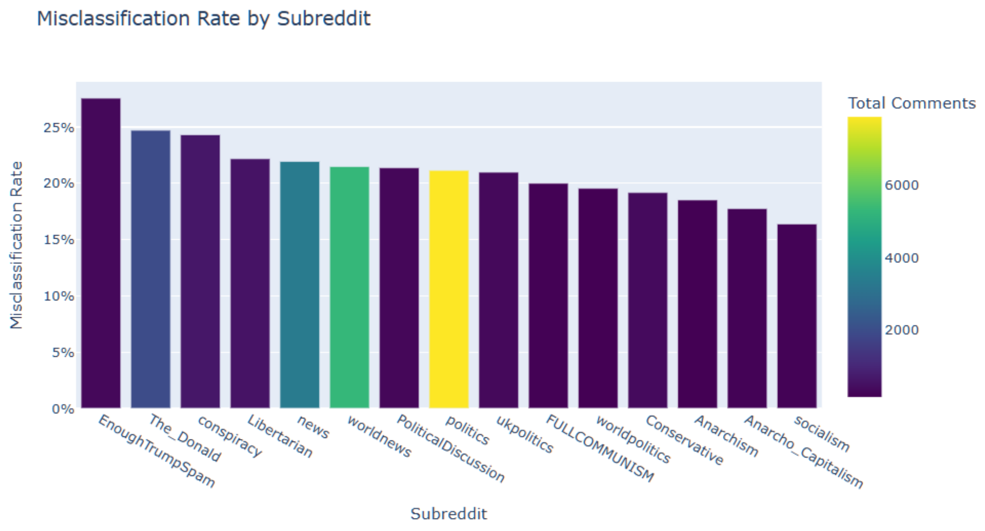


Figure 10: Misclassification Rate by Subreddit for the DistilRoBERTa Model (Top 15 Subreddits by Volume).

Figure 10 illustrates the misclassification rate for the top 15 subreddits ranked by comment volume. A notable variation in error rates is observed, ranging from approximately 16% in `socialism` to over 27% in `EnoughTrumpSpam`. Subreddits such as `EnoughTrumpSpam`, `The_Donald`, and `conspiracy` exhibit higher misclassification rates (above 22%), suggesting that the linguistic styles, specific topics, or types of sarcasm prevalent in these communities might pose greater challenges for the model.

This variance suggests that model generalization across diverse online political communities is not uniform. Factors potentially contributing to higher error rates in certain subreddits could include unique slang, higher prevalence of hyper-partisan language, reliance on specific in-group knowledge for interpreting sarcasm, or perhaps a different distribution of sarcasm types compared to the overall training data. Interestingly, subreddit volume (indicated by color intensity in Figure 10) does not show a simple direct correlation with the error rate; for example, the high-volume `politics` subreddit displays a moderate misclassification rate (around 21%). This highlights the complexity of discourse variation even within the broader category of "political comments" and points towards the potential benefit of incorporating community-specific information or adaptation strategies for improved performance.

3.4 LDA Topic Modeling Results

LDA identified 5 distinct topics each within the sarcastic and non-sarcastic comment sets. The top words for representative topics are listed below (full lists omitted for brevity).

Sarcastic Topics (Examples):

- Topic 1: yes (15.7%), know (13.0%), good (12.4%), ... (e.g., feigned agreement / national critique.)
- Topic 2: like (17.8%), dont (16.0%), racist (6.7%), ... (e.g., disagreement / accusation.)

Non-Sarcastic Topics (Examples):

- Topic 1: news (20.1%), good (10.1%), time (9.9%), ... (e.g., news discussion)
- Topic 2: think (12.9%), love (9.1%), hate (8.6%), ... (e.g., strong opinion / emotional reaction.)

Table 4: Topics Sorted by How Distinctive They are for Sarcasm.

Topic	Sarcastic %	Non-Sarcastic %	Difference
Topic 2	20.376013	17.739013	2.637000
Topic 3	22.310399	21.767862	0.542538
Topic 5	18.030657	17.714144	0.316514
Topic 4	19.069888	20.165559	-1.095671
Topic 1	20.213042	22.613422	-2.400380

Figure 11 compares the overall prevalence of each topic cluster in sarcastic versus non-sarcastic documents.

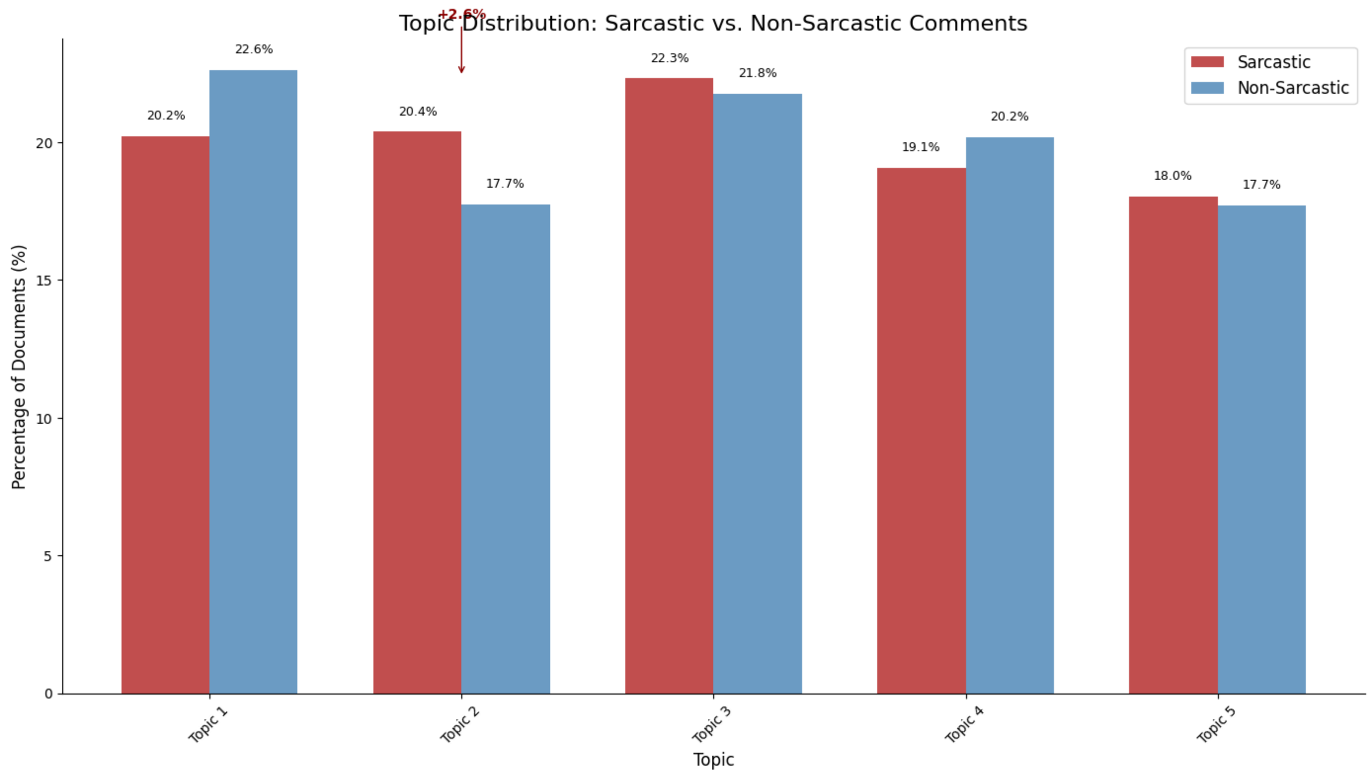


Figure 11: Distribution of Dominant Topics in Sarcastic vs. Non-Sarcastic Comments.

I identified Topic 2 has the largest difference in prevalence between the two sets. Figure ?? compares the word distributions between Sarcastic and Non-Sarcastic comments in Topic 2.

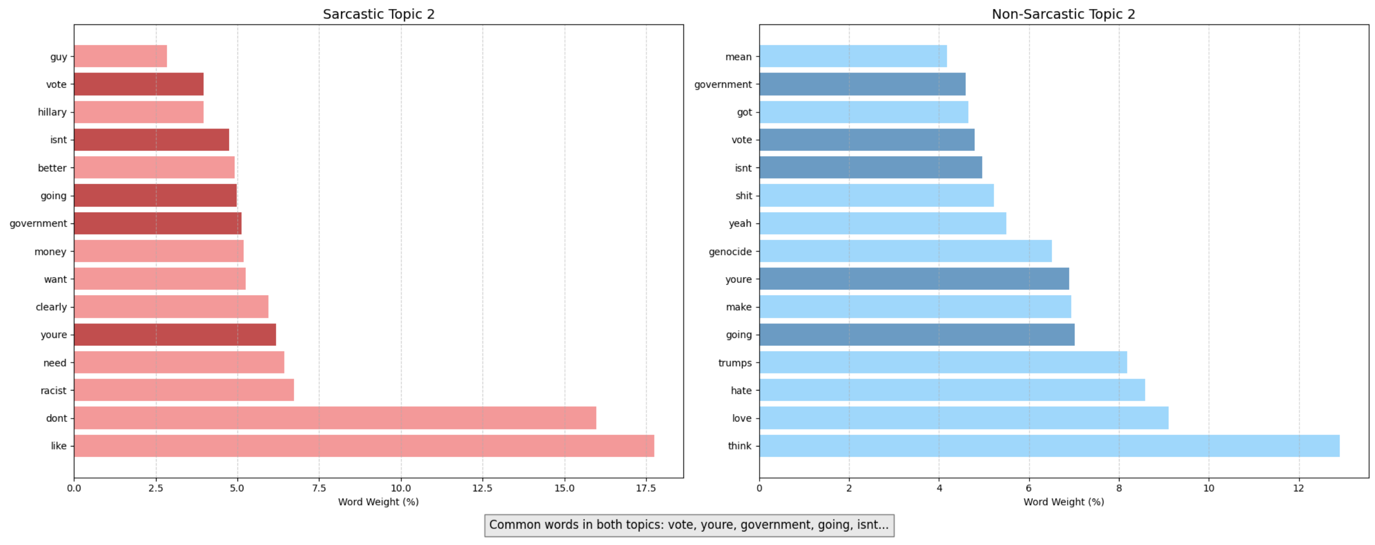


Figure 12: Comparison of Word Weights for Topic 2.

4 Discussion

4.1 Exploration Beyond Baseline: Traditional vs. Transformer

The results clearly indicate that the fine-tuned DistilRoBERTa model significantly outperforms all traditional NLP models evaluated, achieving an F1-score of 0.7814 compared to the best traditional model's 0.7126. This underscores the power of large pre-trained language models to capture complex linguistic patterns, including contextual nuances essential for sarcasm detection, without explicit feature engineering.

However, my exploration with traditional methods revealed the substantial impact of **targeted feature engineering**. Incorporating features related to linguistics, sentiment dynamics, readability, and context consistently improved the performance of BoW, TF-IDF, and N-gram models over using text vectorization alone. Features like sentiment contrast and subreddit sarcasm rate likely provided signals not easily captured by simple N-gram representations. While not reaching transformer-level performance, this demonstrates that well-designed features can significantly enhance simpler models, making them potentially viable in resource-constrained environments.

The **LDA topic modeling** provided a unique perspective beyond classification performance. By analyzing topic differences, I observed distinct thematic patterns. Sarcastic comments appeared more frequently in topics related to direct criticism of national policies and political figures, while non-sarcastic comments dominated topics concerning news report and emotional reaction. This suggests that the *subject matter* itself is correlated with the likelihood of sarcastic expression in this domain.

4.2 Innovation and Creativity

The primary innovation in the traditional NLP pipeline lies in the **breadth and combination of engineered features**. Moving beyond standard punctuation or sentiment scores to include features like sentiment standard deviation across sentences (capturing volatility), subreddit-specific sarcasm rates (context), and specific linguistic markers (intensifiers, contrast words) represents a creative attempt to codify diverse indicators of sarcasm.

For the transformer approach, while fine-tuning is standard, the detailed **interpretability analysis** using a wide range of visualization provides deeper insights than simply reporting performance metrics. This is crucial for understanding the multifaceted results and communicating the findings effectively.

5 Lessons Learned and Potential Improvements

This project yielded several key takeaways and highlighted areas for future work:

Lessons Learned:

- **Transformers Excel but Features Matter:** While DistilRoBERTa offered the best performance, well-crafted features significantly improve traditional methods, suggesting a hybrid approach could be explored.
- **Context is Crucial:** The importance of subreddit-level features (sarcasm rate, error rate analysis) and the nature of LDA topics underscore that political sarcasm is highly contextual. Models likely benefit from incorporating more contextual information (e.g., parent comments, author history, broader thread context).
- **Interpretability Remains a Challenge:** While visualizations help, truly understanding *why* a complex model makes a specific prediction, especially for nuanced phenomena like sarcasm, requires ongoing research into interpretability techniques.
- **'/s' is Imperfect Ground Truth:** Relying solely on '/s' might bias the model towards more explicit sarcasm, potentially missing subtler forms. The high performance might partially reflect learning to detect '/s'-style sarcasm rather than all sarcasm.

Potential Improvements and Future Work:

- **Incorporate Parent Comment Context:** Sarcasm often relies on direct replies. Including the parent comment's text or embedding as input could significantly improve model performance.
- **Explore Different Architectures:** Fine-tuning larger models (RoBERTa-large, DeBERTa) or architectures specifically designed for discourse analysis might yield further gains.
- **Refine Feature Engineering:** Experiment with more sophisticated linguistic features (e.g., dependency parsing, semantic incongruity measures) or interaction features.
- **Improve LDA:** Experiment with different numbers of topics, incorporate metadata into the topic modeling process (e.g., Author-Topic models), or use more advanced topic models.
- **Address Data Bias:** Augment the dataset with examples of sarcasm identified through other means (e.g., human annotation studies) to reduce reliance on '/s'. Explore techniques for detecting implicit sarcasm.
- **Multi-Modal Analysis:** If available, incorporating user flair, post context (images/links), or author characteristics could provide additional signals.
- **More Rigorous User Study:** Conduct the external testing outlined below to get qualitative feedback on model failures and successes in real-world scenarios.

6 Conclusion

This research conducted a thorough investigation into sarcasm detection within the challenging domain of political Reddit comments. I systematically compared traditional NLP methods, augmented by novel feature engineering, against a fine-tuned DistilRoBERTa transformer model. My findings confirm the superior performance of transformer-based approaches for capturing the nuances of sarcasm, achieving a test F1-score of 0.7814. However, I also demonstrated that extensive feature engineering significantly boosts the performance of traditional models, making them more competitive.

Through detailed analysis, including feature importance, attention visualization, token importance, error analysis, and LDA topic modeling, I gained insights into model behavior and the characteristics of political sarcasm on Reddit. LDA revealed distinct thematic differences between sarcastic and non-sarcastic discourse, highlighting topic areas prone to sarcastic expression. The interpretability analyses provided clues about the features and contextual cues leveraged by both types of models.

While achieving high performance on this dataset, I acknowledge the limitations, particularly the reliance on '/s' for labeling and the inherent complexities of textual sarcasm. Future work should focus on incorporating richer context, exploring more advanced model architectures, and validating models through rigorous user studies. This work contributes a detailed comparative analysis and highlights the value of combining strong model architectures with domain-specific insights and interpretability techniques for tackling complex NLP tasks like political sarcasm detection.