b27

sahil_final2.pdf

Vishwakarma Group of Institutions

Document Details

Submission ID
trn:oid:::3618:91655351

Submission Date
Apr 17, 2025, 2:19 PM GMT+5:30

Download Date
Apr 17, 2025, 2:22 PM GMT+5:30

File Name
sahil_final2.pdf

File Size
648.5 KB

9 Pages

4,302 Words

25,071 Characters

0% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Detection Groups

1 AI-generated only 0%
Likely AI-generated text from a large-language model.

2 AI-generated text that was AI-paraphrased 0%
Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

What does 'qualifying text' mean?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be likely AI-generated will be highlighted in cyan in the submission, and likely AI-generated and then likely AI-paraphrased will be highlighted purple.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

Sarcasm Sleuth: A Comparative Study of Transformer and Machine Learning Models for Sarcasm Detection on the MUStARD Dataset

Shivam Shailesh Wangikar
Dept. of Computer Science & Engineering (AI)
BRACT's Vishwakarma Institute of Information Technology,
Pune, India
shivam.22311287@viit.ac.in

Gaurav Mangesh Yadav
Dept. of Computer Science & Engineering (AI)
BRACT's Vishwakarma Institute of Information Technology,
Pune, India
mangesh.22311296@viit.ac.in

Sahil C. Bhagwat
Dept. of Computer Science & Engineering (AI)
BRACT's Vishwakarma Institute of Information Technology,
Pune, India
sahil.22311046@viit.ac.in

Apurva Pramod Jangle
Dept. of Computer Science & Engineering (AI)
BRACT's Vishwakarma Institute of Information Technology,
Pune, India
apurva.22311433@viit.ac.in

***Abstract*—The task of detecting sarcasm, especially when it occurs in the conversational setting, is quite a challenging task for natural language processing. The performance of various machine learning models i.e. variations of Support Vector Machine like Linear Kernel & RBF Kernel, and state of art deep learning BERT, RoBERTa and XLNet for sarcasm detection from the MUStARD data set is quantified. The television shows of which a dataset of the conversational utterances is made, allow for a rich context for the task of recognizing sarcasm. In this work we try to understand what is the impact of fine-tuning, feature engineering and contextual embeddings to the model performance. It is shown that, optimized with cross validation and hyperparameter tuning, RoBERTa achieves best performance in sarcasm classification. Also, the recall is further enhanced with the use of linguistic and sentiment-based features. Precision of traditional models such as Support Vector Machines is very good but it has poor recall due to the lack of methods for rule-based feature extraction for sarcasm detection. Results indicate the superiority of transformer models on classical approaches for the understanding of conversational**

sarcasm. We discuss the way the data can be integrated with other modalities, the further development of context models and the possibility to explore more feature engineering techniques in order to improve sarcasm detection in textual communication in future research.

Keywords— *Sarcasm Detection, Natural Language Processing, Bidirectional Encoder Representations from Transformers (BERT), Robustly Optimized BERT Pretraining Approach (RoBERTa), Support Vector Machine (SVM), Generalized Autoregressive Pretraining for Language Understanding (XLNet), Multimodal Sarcasm Detection Dataset (MUStARD), Comparative Study, Transformer-based Models, Text Classification.*

I. INTRODUCTION

Verbal irony in the form of sarcasm is particularly hard to handle by Natural Language Processing (NLP) systems [1]. Typically for its detection we have to know not only what words mean literally but also how to use them in context, their intended meaning, speaker tone and sometimes – but usually not – facial expressions or gesture. For applications such as the sentiment analysis, opinion mining, and content moderation online that involve misinterpreting of sarcastic remarks can result in invalid conclusions, it is crucial to automate the detection of sarcasm.

Over the last couple of years, NLP has made impressive progress, especially with the appearance of large pretrained Transformer models such as BERT [2], RoBERTa [3], and XLNet [4], that have, among other things, achieved great success across a wide selection of language understanding tasks. Self-attention mechanisms allow these models to capture the long-range dependencies and the contextual nuances in a text. However, their effectiveness specifically for sarcasm detection, especially in conversational settings, warrants comparative investigation against both each other and more traditional machine learning approaches like Support Vector Machines (SVM) [5]. Text classification tasks have been often solved by SVMs with feature engineering techniques such as TF-IDF.

In particular, this study empirically compares BERT, RoBERTa, XLNet, and SVM on the performance for sarcasm detection using the Multimodal Sarcasm Detection Dataset (MUStARD) [6]. That is to say, it takes the cue from popular TV shows (it is substantial in The Big Bang Theory, Sarcasmoholics, Friends, The Golden Girls, as mentioned in our EDA) and provides conversational utterances in the context of its previous instances making it apt for evaluating context aware sarcasm detection. The dataset that we analyzed in our EDA had 690 samples and were perfectly balanced between the sarcastic and not sarcastic labels (345 of each). MUStARD is a multimodal data (text, audio, video) and “Sarcasm Sleuth” is only on textual modality, if we evaluate the performance of better models at ruling out sarcasm from utterances and their textual context.

To do so, we implemented by BERT, RoBERTa as well as XLNet using the Hugging Face Transformers library [7]. Roberta was explored with specific variations to see the impact of the feature engineering (with sentiment and linguistic features) as well as context embedding as well as the effect of hyperparameter tuning using 3-fold cross validation. scikit-learn [8] was used for training SVM models with TF-IDF features both using Linear and RBF kernels. The experiments were made in Google Colab.

In this paper we make the main contribution of a direct, empirical comparison of these disparate models to the sufficiently large MUStARD text-based sarcasm detection dataset. Finally, we provide quantitative results with the help of standard metrics (accuracy, precision, recall, F1-score) and discuss insights taken from model performance and EDA – strengths and weaknesses of each approach in identifying conversational sarcasm.

The rest of this paper will proceed as follows: Section II covers related work in sarcasm detection. Partial description of methodology is presented in Section III: dataset description, preprocessing steps, model architectures and experimental setup. The results include performance metrics, and a visualization summary is provided in Section IV. In Section V the findings are discussed, limitations, and comparisons are done. Finally, the paper concludes in Section VI by suggesting future research directions.

II. RELATED WORK

The research of sarcasm detection is an active NLP research area for some time. Early solutions rely on the rule based or lexicon-based methods that could be based on searching the specific cues such as little interjections, positive verbs with negative statements, or an explicit marker [9]. While useful in some cases, sarcasm is a notoriously hard task to learn due to the implicit and context dependent nature of the phenomenon. Most of the stuff have been applied by the machine learning techniques, although that is SVMs. The works like [10] used n-grams, sentiment lexicons and syntactic features with SVMs to identify sarcastic tweets. In these traditional approaches, feature engineering is extremely important and requires domain expertise to identify relative indicators of sarcasm. We explored such deep learning methods, more recently, and our work then encompassed SVM models (Linear and RBF kernel) of TF-IDF features as a baseline comparison. With the rise of deep learning, lots of architectures of neural network have been explored. Text was used to automatically learn features involving RNN (Recurrent Neural Networks) and CNN (Convolutional Neural Networks) particularly in the case of the LSTM. for sarcasm detection [11]. These models performed better than traditional methods by considering this sequential information.

It was the advent of Transformer models which made a huge leap of progress. In fact, BERT [2] and other related models (ROBERTa [3], XLNet [4]) fine-tuned from large text corpora achieve the first results remarkable in their state. There have been several uses of Transformers to detect sarcasm. As an example, BERT was shown to be effective in [12], which also used BERT to perform better than other RNN and CNN based models in benchmark datasets. There has been some work trying to make it more explicit use of context within Transformer architectures for conversational sarcasm [13].

Often, multimodal approaches with audio and visual cues are used in conjunction with text by research using the MUStARD dataset [6]. Some of these studies show the need for sarcasm detection using the non-textual information [14].

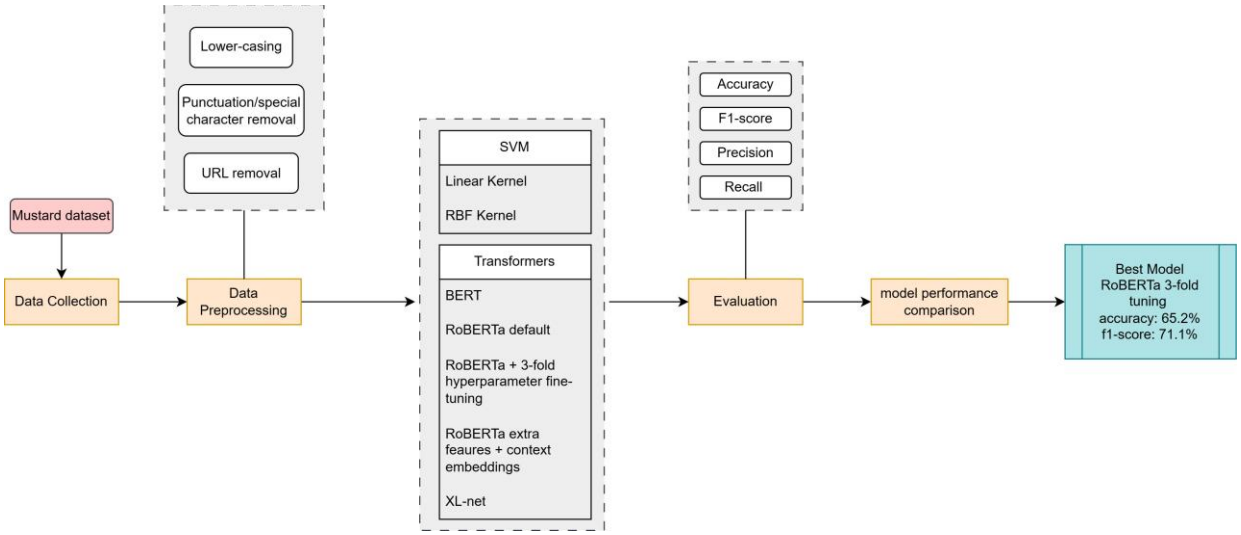

Fig. 1. Sarcasm Detection Methodology Flowchart

However, to understand what performance we can achieve with textual data alone is very important in situations where it is not possible to have multimodal data.

We make a very focused and direct comparison in this field by only employing the textual data of the MUStARD dataset for Transformer models (BERT, RoBERTa, XLNet) and SVMs. To that end, in this work we analyze the advantages of fine-tuning strategies, context embedding and feature engineering in the RoBERTa framework, and illustrate how one could improve Transformer performance for conversational sarcasm detection in that particular dataset. We implement and evaluate these particular models under a common experimental setup on MUStARD, whereas broader surveys do not.

III. METHODOLOGY

In this section, dataset is described, the preprocessing techniques, the used machine learning models and setup of experiments are described in the Sarcasm Sleuth project. The overall methodology is depicted in Fig. 1.

A. Dataset

Only MUStARD [6] dataset is used in the study. MUStARD is composed of video clips, audio clips and text data from American TV sitcoms The Big Bang Theory (BBT), Friends, The Golden Girls and the web series Sarcasmoholics, from which only text data is used in this research. An utterance, the preceding conversational context (as text), the speaker, show name and a binary label for whether the utterance is sarcastic (True/1) or not (False/0) is given as a single data point.

We carry out EDA using pandas in Google Colab and we found the dataset contains 690 samples. The dataset is perfectly balanced with 345 sarcastic and 345 non-sarcastic instances, as shown in Fig. 2. For this project, we only utilize the textual part of this data, namely utterance and context, for training and evaluation of the model. Exploratory analysis also revealed distributions of utterance length (Fig. 3) and context length (Fig. 4). Further EDA insights are presented below (Figs. 5 to 9).

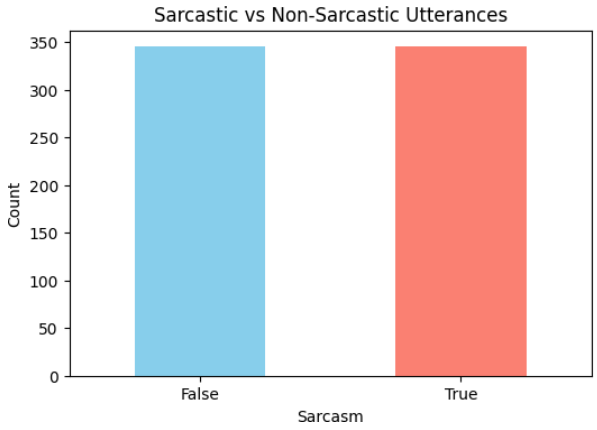

Fig. 2. Distribution of Sarcastic vs Non-Sarcastic Utterances in the MUStARD dataset.

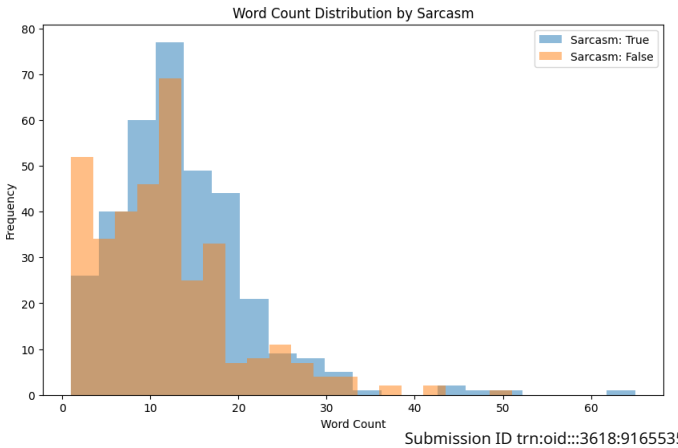

Fig. 3. Word Count Distribution per Utterance by Sarcasm Label

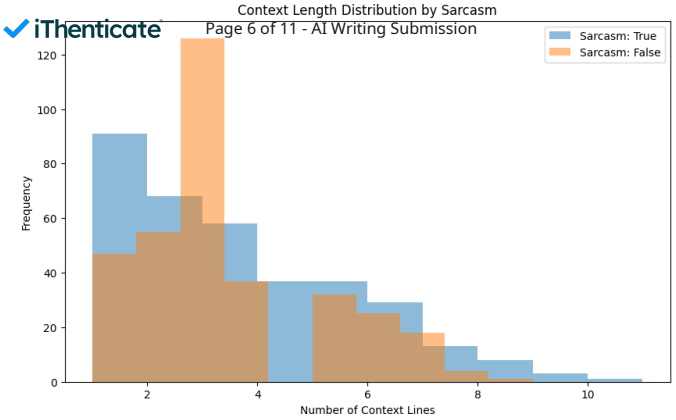

Fig. 4. Distribution of the Number of Context Lines by Sarcasm Label.

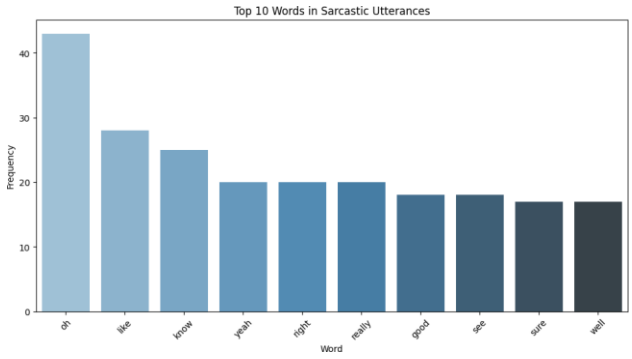

Fig. 5. Top 10 most frequent words in sarcastic utterances (after preprocessing).

Fig. 6. Word clouds for sarcastic (left) and non-sarcastic (right) utterances.

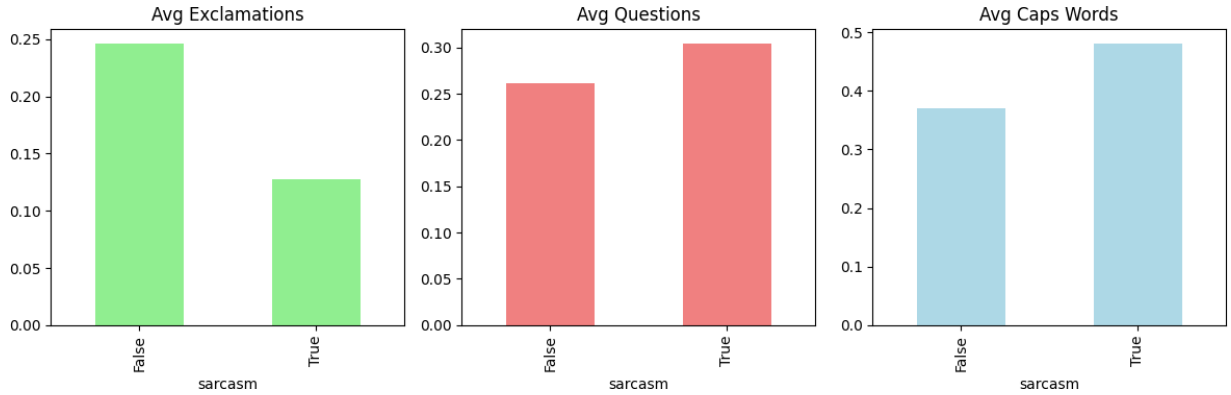

Fig. 7. Analysis of average exclamation marks, question marks, and capitalized words per utterance by sarcasm label.

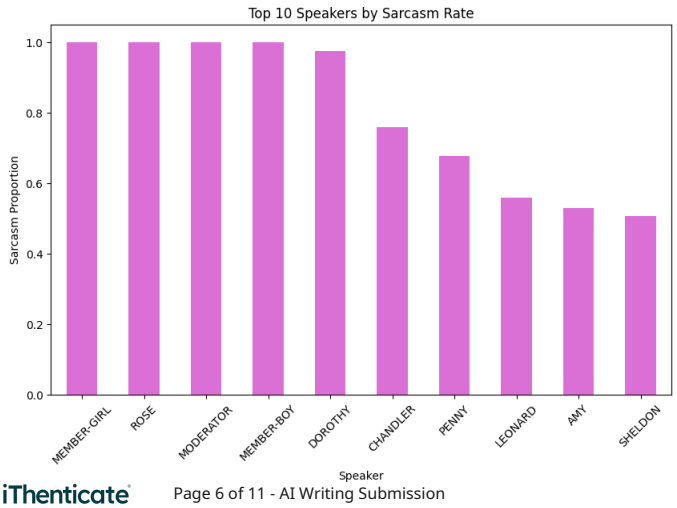

Fig. 8. Top 10 speakers ranked by their sarcasm rate in the dataset.

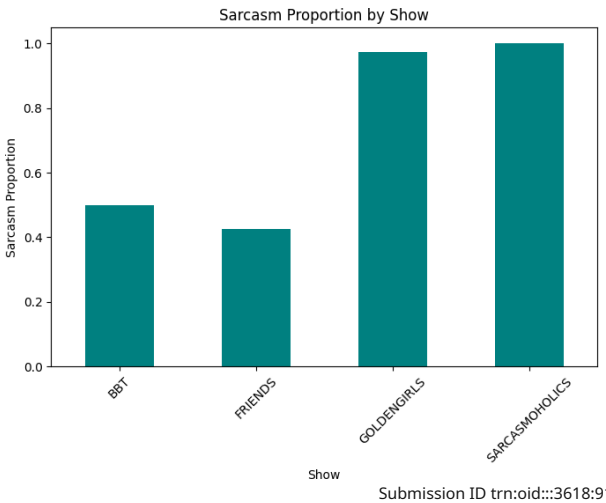

Fig. 9. Sarcasm proportion across different TV shows in the dataset.

This study uses a methodology in which the data preprocessing to model validation is illustrated in Fig. 1. The balance in the MUStARD dataset is shown in Fig. 2, which is a perfectly balanced class distribution. Fig. 3 and Fig. 4 show some patterns within utterance length and context lines distinguishing sarcasm from no sarcasm. Fig. 5 and Fig. 6 presents frequent words and word clouds, providing lexical differences. Fig. 7 describes the expressive patterns revealed in punctuation usage in sarcastic speech. Fig. 8 and Fig. 9 present speaker-level sarcasm trends and variations across TV shows.

B. Preprocessing

Textual data (utterances and context turns) required a number of preprocessing steps following findings from EDA:

- **Lowercasing:** All text were converted to lowercase.
- **Remove URL and Mention:** URL (Starts with 'http' or 'www.') and user mentions (which start with '@user-name') were stripped out via regular expressions. Hashtags were removed but the word itself was preserved (e.g., '#sarcasm' became 'sarcasm').
- **Punctuation and Special Character Removal:** All punctuation marks along with non-alphanumeric characters (except for spaces) were removed. It also filtered out non-ASCII characters.
- **Whitespace Stripping:** Consecutive whitespace characters were condensed to single spaces and leading/trailing whitespace was stripped.
- **Stop Word Removal:** The NLTK library [15] was used to get rid of common English stop words (e.g., "the", "a", "is"). Here, this was mostly used prior to TF-IDF vectorization for SVMs.
- **Lemmatization:** (Used only in the SVM notebook) The WordNet Lemmatizer from NLTK was used to reduce the words to their base or dictionary form.
- **Tokenization:** Input text was tokenized into sequences that the model could understand. For transformer models, computing simple specialized tokenizers (e.g., "Roberta-Tokenizer', 'BertTokenizer', 'XL-NetTokenizer' through 'AutoTokenizer') from the Hugging Face library were employed, doing subword tokenization (i.e., WordPiece or BPE) and particular tokens inclusion ([CLS], [SEP]). These tokenizers also dealt with padding/truncating up to a fixed maximum length (for example, 128 tokens). Tokenizing was implicitly part of the TF-IDF vectorization process (after the first text cleaning) for SVMs.

Context was considered the concatenation of previous utterances, when necessary (i.e., for context embeddings) was joined before tokenization.

C. Models

Thus, four main types of models were implemented and compared.

1) **BERT:** We used pre-trained 'bert-base-uncased' model from Hugging Face Transformers [2]. On top of the pooled output of the base model, a sequence classification head (a linear layer) was added. Finally, the entire model was fine-tuned on the binary sarcasm classification dataset MUStARD.
2) **RoBERTa:** The 2nd: 'roberta-base' model served as a foundation [3]. We evaluated three variations:
   a) As BERT, a base RoBERTa model with a classification head was fine-tuned on the task.
   b) Extra Features & Context Embedding + RoBERTa: The input was augmented with this variant. The context was tokenized and passed through the RoBERTa base model without the classification head, then the mean of the last hidden state across all tokens was computed and taken as the context embedding. Given that these context embeddings were most probably concatenated or otherwise combined with the utterance embeddings, unlikely there could be so large difference between the sum of possible feature values for these embeddings, unless the concat was accompanied by a learnability of the resulting concatenated layer. Additional input to the classifier may have been (word count, uppercase count, average word length, lexical diversity) and sentiment scores calculated using TextBlob [16] on the utterance.
   c) Fine tuning the roberta base model on the training data with 3-Fold hyperparameters tuning. Consequently, we experimented with the hyperparameters of learning rate(3e-5) and weight decay (0.02),were adjusted based on the evaluation loss and using early stopping(patience=3), which prevents overfitting and loads the best model per fold. The reported final metrics for this variant are likely the performance on a held-out test set with training on the full training set with optimal hyperparameters or an average across folds if evaluated on that (the report mentions "Final" Model Score).

3) **SVM:** Support Vector Machines [5] were used with scikit-learn [8]. Before vectorizing the numerical features, TF-IDF (Term Frequency-Inverse Document Frequency) was used to vectorize the text data and lemmatize and preprocess the data. Vocabulary size was set to the top 5000 features. Two types of kernels were tried:
   - **Linear Kernel:** A regular linear SVM ('SVC(kernel= 'linear)').
   - **RBF Kernel:** Support Vector Machine with Radial Basis Function kernel ('SVC(kernel= 'rbf')'). Default parameters (C=1.0, gamma= 'scale') were applied as defined in the notebook.
4) **XLNet:** The 'xlnet-base-cased' [4] model was used. Similar to BERT and RoBERTa, the model was fine-tuned for sequence classification on the MUStARD dataset using the Transformers library. As opposed to the masked language model training goal of BERT, XLNet uses a permutation-based training objective.

D. Experimental Setup

Python was used for all model implementations, training, and evaluations in Google Colab notebooks. Among the important libraries were:

- Hugging Face Transformers [7] for the 'Trainer' API for fine-tuning and for accessing pre-trained models (BERT, RoBERTa, XLNet) and their tokenizers.
- PyTorch (which the Transformers "Trainer" uses implicitly).
- For SVM implementation, use scikit-learn [8], TF train-test splitting, IDF vectorization, and evaluation metrics (classification report, F1-score, confusion matrix, accuracy, precision, and recall).
- Pandas is used to load and manipulate data.
- For preprocessing tasks like lemmatization and stop word removal, used NLTK [15].
- Seaborn and Matplotlib are used to create visualizations.
- For sentiment analysis features, see TextBlob [16].
- During the RoBERTa 3-fold tuning procedure, logging was done using Weights & Biases (wandb).

On average we split the dataset with 80% training set and 20% validation/test set (random state=42 used consistently). During cross validation the training set for the 3-fold RoBERTa was also split. The fine-tuned standard hyperparameters were learning rates (unless otherwise specified or tuned) from 2e-5 to 5e-5, batch sizes of 8 or 16, and generally three training epochs as seen in the given notebooks. It was evaluated in terms of accuracy, recall, precision, and F1-score on positive class (sarcasm=1). Also, error patterns were analyzed using confusion matrices.

IV. RESULTS

In this section, we perform the evaluation of the implemented models on the sarcasm data of the MUStARD dataset.

A. Performance Metrics

The result of BERT, RoBERTa (alongwith variations), SVM (Linear and RBF kernels), and XLNet was evaluated using accuracy, precision, recall, and F1-score on the held-out test set (or averaged across folds where applicable). The results are summarized in Table I. Important conclusions from the results table are:

- The 3-fold cross-validation with tuned hyperparameters of the RoBERTa model had the best accuracy (65.2%) and F1-score (71.1%).
- Incorporating RoBERTa with extra features (linguistic, sentiment) and context embeddings also achieved a high F1-score (~71.1%), greatly improving recall to ~75.3% compared to the baseline RoBERTa, albeit with considerably lower accuracy than the tuned one.
- The baseline fine-tuned RoBERTa did not work well, having the worst recall (50.59%) and second-worst F1-score (57.33%).
- SVM using Linear kernel attained comparable accuracy (~63.0%) and high precision (80.0%) but poor recall (53.0%).
- SVM with RBF kernel had extremely high precision (86.0%) but the lowest recall (45.0%).
- BERT and XLNet performed at mid-level, with F1-scores of 64.47% and 60.29, respectively. Although XLNet performed evenly on precision and recall, it failed to outperform BERT or the best-performing RoBERTa variants.

B. Visualization Summary

The performance of the model was compared by visualization and the nature of the dataset was explored (see Figs. 2-9 for EDA visualizations presented earlier, and Figs. 10-12 for performance comparisons).

- **Performance Comparison Plots:** To have a visual comparison of accuracy, recall, precision, and F1 score with all the models and variations we performed, we plotted bar plots (Fig. 10) and Radar plots (Fig. 11). The plots clearly demonstrate performance improvements of the optimized RoBERTa variant and the SVM precision-recall trade-offs. The correlation heatmap (Fig. 12) shows moderate positive correlation between accuracy and F1, recall and precision, which aligns with the observed performance, particularly for SVM.
- **Word Clouds:** Word clouds (Fig. 6) showed that after removal of stop words, the word patterns differed between non-sarcastic and sarcastic sentences. Sarcastic utterances frequently featured words like 'oh', 'like', 'know', 'yeah', 'right', 'really', while non-sarcastic sentences had words like 'oh', 'yeah', 'well', 'go', 'know'. This overlap highlights the context-dependent nature of these interjections. The most frequent words specifically in sarcastic utterances are shown in Fig. 5.
- **Text Length Distribution:** As seen in Fig. 3, sarcastic utterances were slightly longer on average (mean length: 13.5 vs 11.4 for non-sarcastic), suggesting sarcasm might sometimes involve more elaborate phrasing, though the distributions overlap significantly.

TABLE I
OVERALL EVALUATION SUMMARY

Model / Variant	Accuracy	Precision	Recall	F1-Score
RoBERTa (Default)	~53.6%	66.15%	50.59%	57.33%
RoBERTa + 3-Fold + Hyperparameter Fine-tuning	65.2%	72.8%	69.4%	71.1%
RoBERTa + Extra Features + Context Embedding	~62.3%	~67.4%	~75.3%	~71.1%
BERT	~60.9%	67.12%	62.03%	64.47%
SVM (Linear Kernel)	~63.0%	80.0%	53.0%	~64.0%
SVM (RBF Kernel)	~61.6%	86.0%	45.0%	~59.0%
XLNet	~60.9%	61.19%	59.42%	60.29%

Note: Values are approximate where indicated (~) in the source report. The RoBERTa + 3-Fold tuning shows the 'Final' model performance.

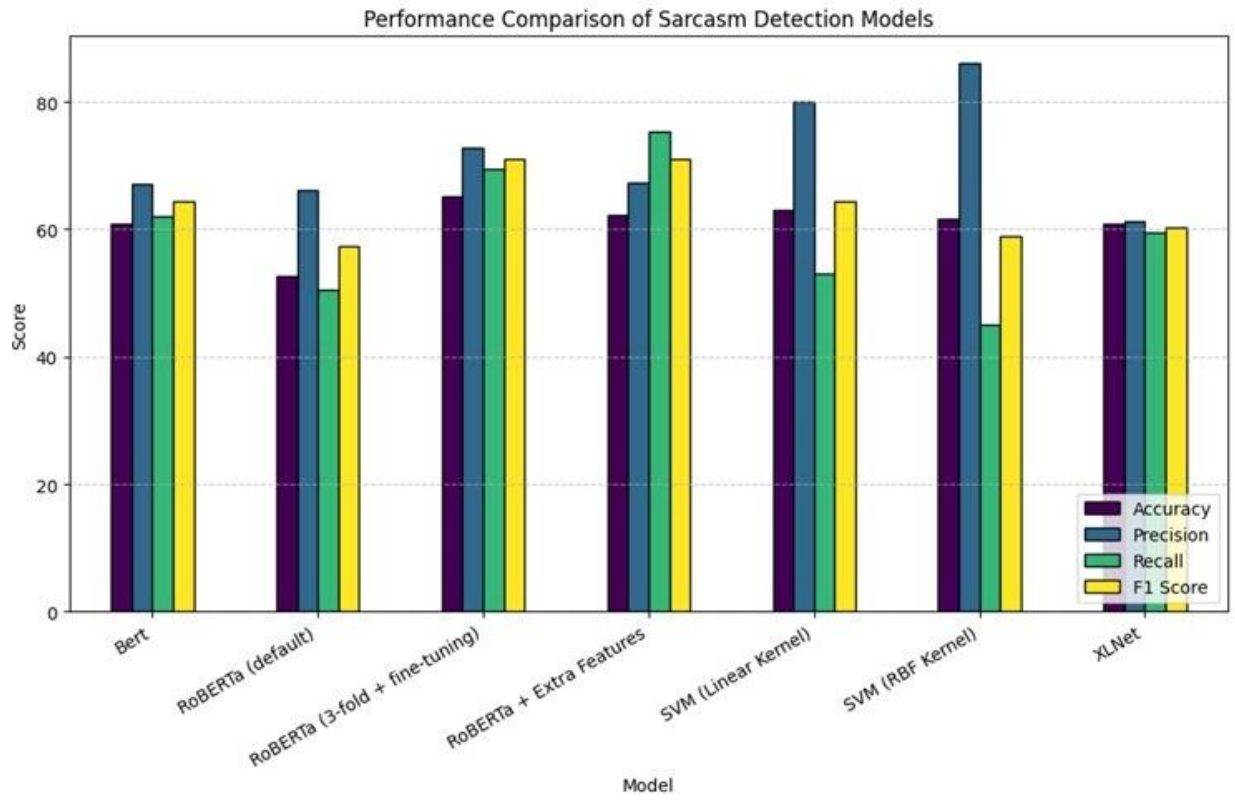

Fig. 10. Comparison of model performance metrics.

- **Class Distribution:** The class proportion of the MUStARD dataset was verified to be perfectly balanced (Fig. 2), with 345 sarcastic and 345 non-sarcastic examples.
- **Other EDA Insights:** Further analysis explored punctuation usage (Fig. 7), speaker tendencies (Fig. 8), and variations across TV shows (Fig. 9), providing additional context on the dataset characteristics.

V. DISCUSSION

The results provide insights into the performance of various models on the conversational MUStARD sarcasm detection task. The RoBERTa model, after 3-fold hyperparameter fine-tuning, achieved the superior performance (Acc. 65.2%, F1 71.1%), underscoring the importance of model tuning, especially for Transformer models on smaller, specialized datasets like MUStARD. Standard, out-of-the-box fine-tuning proved insufficient for this nuanced task, as evidenced by the poor performance of the default RoBERTa baseline (57.3 F1). The tuned hyperparameters (e.g., learning rate 3e-5, weight decay 0.02) were clearly more suitable for this dataset.

Interestingly, the RoBERTa variant augmented with extra features (linguistic/sentiment) and context embeddings achieved a comparable F1 score (~71.1%) but with significantly better recall (~75.3%). This highlights the value of incorporating explicit contextual and linguistic cues for detecting sarcasm. These features likely helped capture subtleties missed by the base Transformer, particularly in identifying sarcastic instances (hence the improved recall). The use of mean-pooled context embeddings allowed condensed information from preceding turns to be leveraged.

SVM models, particularly with a Linear kernel, achieved

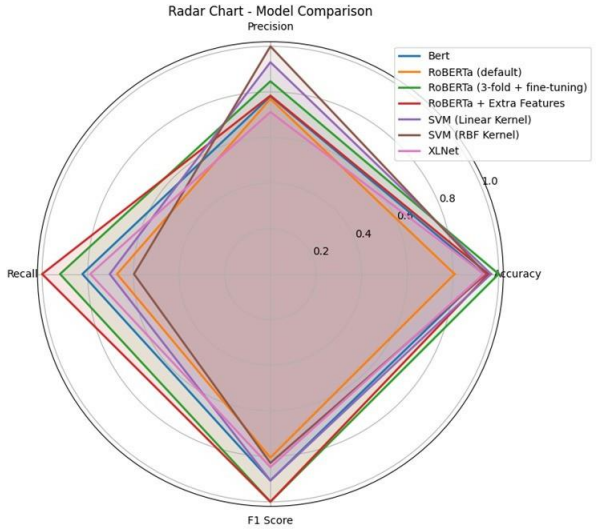

Fig. 11. Radar chart illustrating multi-metric model comparison.

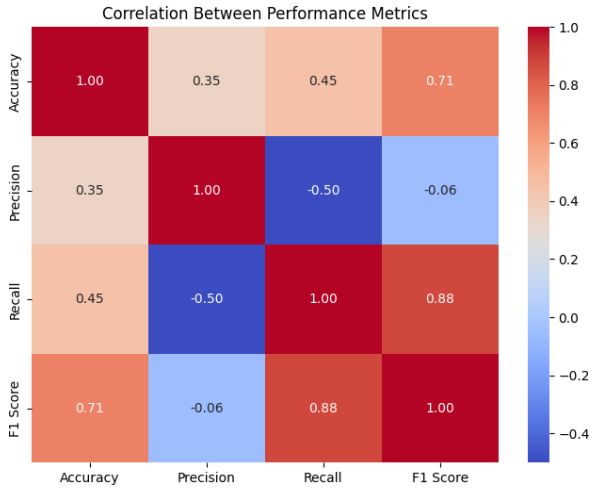

Fig. 12. Correlation between performance metrics across models.

reasonable accuracy (~63.0%) and strong precision (80-86%). This indicates that TF-IDF features can capture some discriminating signals for sarcasm. However, their significantly lower recall (45-53%) compared to the best RoBERTa variant resulted in lower F1 scores. This limitation likely stems from TF-IDF's inability to capture word order, long-range dependencies, and contextual nuances as effectively as Transformers. The RBF kernel's extremely high precision coupled with very low recall suggests it identifies only the most unambiguous sarcastic examples, failing on more subtle cases.

BERT and XLNet performed moderately, with F1 scores slightly below the linear SVM. While their performance is consistent with other studies on this dataset, it suggests RoBERTa's training objective or architecture might be slightly advantageous here. Alternatively, BERT and XLNet might also benefit from similar hyperparameter tuning or feature augmentation.

The EDA findings align with the task's challenges. The frequent overlap in common words ("oh", "yeah", "know") between sarcastic and non-sarcastic utterances (Fig. 6, Fig. 5) necessitates models that look beyond individual words and utilize context. The slightly longer average length of sarcastic utterances (Fig. 3) is likely too weak a signal alone. The importance of conversational context, explicitly provided in MUStARD (distribution shown in Fig. 4), is reinforced by the success of the feature-enhanced RoBERTa variant which incorporated context embeddings.

Limitations of this study include the comparably small sample size (690 instances) in MUStARD dataset, which might affect the generalization ability of models, based on deep learning, and make them sensitive to hyperparameter choices. Furthermore, this study focused solely on textual data, omitting potentially valuable audio and visual cues present in the original multi-modal dataset. A more exhaustive hyperparameter search for the tuned RoBERTa, perhaps beyond the 3-fold CV, might yield further improvements. Finally, exploring the same feature engineering techniques on BERT and XLNet could provide a more complete comparison.

VI. CONCLUSION

The paper 'Sarcasm Sleuth' presents a comparative analysis of BERT, RoBERTa, XLNet, and SVM models for text-based sarcasm detection on the conversational MUStARD dataset. Our findings demonstrate that well-tuned Transformer models, particularly RoBERTa augmented with relevant features or optimized via cross-validated hyperparameter tuning, significantly outperform traditional SVM methods for this task.

Specifically, RoBERTa fine-tuned with optimized hyperparameters achieved the highest F1 score (71.1%) and accuracy (65.2%). Incorporating extra linguistic/sentiment features and context embeddings into RoBERTa yielded a similar F1 score but notably improved recall, emphasizing the importance of context and explicit cues. While SVMs showed good precision, their recall was limited. BERT and XLNet performance was moderate.

The results indicate that standard fine-tuning is often suboptimal for complex tasks like conversational sarcasm detection on specialized datasets. Cross-validated hyperparameter optimization or strategic feature/context augmentation is crucial for unlocking the potential of Transformer models. Future work could explore the multimodal aspects of the MUStARD dataset by incorporating audio and visual features. Additionally, investigating more sophisticated context integration methods within Transformer architectures, applying similar feature engineering to BERT/XLNet, using larger datasets, and performing detailed error analysis could further advance the understanding and detection of conversational sarcasm. Ensemble methods combining the strengths of different models also represent a promising direction.

REFERENCES

[1] R. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," in *Proceedings of the ACL-*

02 conference on Empirical methods in natural language processing - Volume 10, 2002, pp. 79-86.

[2] Devlin, J. et al. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT* (pp. 4171–4186).

[3] Liu, Y. et al. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint*, arXiv:1907.11692.

[4] Yang, Z. et al. (2019). XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, 32.

[5] Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.

[6] Castro, S. et al. (2019). MUStARD: A multimodal sarcasm detection dataset. In *Proceedings of ACL* (pp. 2409–2419).

[7] Wolf, T. et al. (2020). Transformers: State-of-the-art natural language processing. In *EMNLP (System Demonstrations)* (pp. 38–45).

[8] Pedregosa, F. et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.

[9] Riloff, E. et al. (2013). Sarcasm as contrast between positive sentiment and negative situation. In *EMNLP* (pp. 704–714).

[10] Davidov, D., Tsur, O., & Rappoport, A. (2010). Semi-supervised recognition of sarcastic sentences in Twitter and Amazon. In *Proceedings of CoNLL* (pp. 107–116).

[11] Poria, S. et al. (2016). A deeper look into sarcastic tweets using deep CNNs. In *Proceedings of COLING* (pp. 2700–2710).

[12] Ghosh, D. & Muresan, W. (2018). Sarcasm analysis using conversational context. *Computational Linguistics*, 44(4), 755–792.

[13] Potamias, I., Siolas, K., & Stafylopatis, A. (2021). Transformer-based sarcasm detection in conversation threads. In *Proceedings of EACL* (pp. 59–69).

[14] Hazarika, D., Zimmermann, R., & Poria, S. (2020). MISA: Modality-invariant and specific representations for multimodal sentiment analysis. In *Proceedings of ACM Multimedia* (pp. 1122–1131).

[15] Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly Media.

[16] Loria, S. (2018). *TextBlob Documentation*. Retrieved from https://textblob.readthedocs.io/