Sarcasm Detection Final Report

Adnan Al Joubi, Mitashi Parikh, Swara Kurakula, Yayun Tan

Introduction

In this project, we developed a sarcasm detection system for news headlines using a variety of deep learning models, including logistic regression, SVM, LSTM, BiLSTM, and BERT. Sarcasm, a form of verbal irony in which the intended meaning is often the opposite of the literal message, can flip the sentiment of a statement. Combined with its implicit nature and reliance on contextual cues this makes sarcasm detection a particularly interesting yet challenging problem to solve, complicating other NLP tasks such as sentiment analysis and opinion mining.

Our approach involved data preprocessing, such as converting headlines to lowercase, removing punctuation, tokenizing, and padding sequences, followed by an iterative cycle of model development and refinement. Starting with baseline models and progressively integrating advanced techniques like pretrained GloVe embeddings and regularization strategies (i.e. dropout, weight decay, and early stopping), we aimed to enhance the models' generalization capabilities and effectively address the complexities of sarcasm detection in natural language.

Background

Gregory et al. (2020) in *A Transformer Approach to Contextual Sarcasm Detection in Twitter* explored sarcasm classification in tweets using an ensemble of transformer models, including BERT, RoBERTa, XLNet, and ALBERT, demonstrating that transformer-based approaches outperform LSTM and GRU architectures in detecting sarcasm within social media conversations.

In the paper *On Sarcasm Detection with OpenAI GPT-based Models*, the authors explore the application of Generative Pretrained Transformer (GPT) models for sarcasm detection in text. They evaluate various GPT versions, including GPT-3, InstructGPT, GPT-3.5, and GPT-4 and highlight the variability in sarcasm detection performance across different GPT model releases.

The GitHub repository, *Sarcasm-Detection*, is an academic project focused on detecting sarcasm in tweets using both traditional machine learning (SVMs, logistic regression) and deep learning models (CNNs, LSTMs, GRUs, BiLSTMs, and attention-based LSTMs). The project evaluates these models on four Twitter datasets.

Data and Methods

For the data, we used a publicly available dataset from Kaggle, which consists of approximately 27,000 news headlines. In the data, sarcastic headlines were sourced from The Onion, while non-sarcastic headlines were taken from The Huffington Post. Various model types were implemented and improved upon.

Traditional Machine Learning Models (SVM+Logistic Regression)

For sarcasm detection, Logistic Regression and Support Vector Machines (SVM) were trained using TF-IDF features extracted from preprocessed headlines. The dataset was split into an 80% training set and a 20% test set, with text cleaning involving lowercasing, punctuation removal, lemmatization, and stopword filtering. Logistic Regression, a linear classifier using the sigmoid function, was trained with max_iter=200 to ensure convergence. SVM, employing a linear kernel, sought to maximize the margin between classes. Both models relied on n-gram-based features, lacking deep contextual understanding, and exhibited a tendency to overfit due to the high dimensionality of text data.

LSTM

For the LSTM experiments, headlines were lowercased, punctuation was removed, and the text was tokenized. A vocabulary was built from the training split with special tokens <PAD> and <UNK>, and each headline was converted into a fixed-length sequence via padding/truncation (max length = 20). The dataset was split into train/test (80/20), and a validation split was taken from the training set (10%) for early stopping. Training used PyTorch DataLoaders for batching and shuffling.

We compared two models: (1) a baseline LSTM with randomly initialized embeddings, and (2) an improved LSTM initialized with GloVe-100d embeddings (fine-tuned during training). Both models used an LSTM with 2 layers and hidden size 128, plus dropout (0.2) and early stopping (patience 3). Optimization used Adam (lr=1e-3); the improved model additionally used weight decay (1e-7).

BiLSTM

To train the BiLSTM model for sarcasm detection, we split the dataset into training, validation, and test sets to evaluate generalization. We preprocessed the text by tokenizing headlines, padding sequences to a uniform length, and embedding words using pre-trained GloVe vectors. Our final model consisted of an embedding layer initialized with GloVe embeddings (trainable), three bidirectional LSTM layers with 8 units each, dropout layers (0.5) to prevent overfitting, and a dense layer with a sigmoid activation function for binary classification. We addressed overfitting by tuning dropout rates, using a small number of LSTM units due to the dataset size, and optimizing the learning rate (0.0005) with the Adam optimizer. Additionally, we hypertuned hyperparameters such as LSTM unit size, etc.

BERT

The dataset was split into training, development, and test sets to enable model training, hyperparameter tuning, and unbiased evaluation. BERT (bert-base-uncased) was chosen for its bidirectional attention mechanism, which dynamically adjusts word meanings based on context—essential for detecting sarcasm's implicit contradictions and tonal shifts. Preprocessing involved tokenizing headlines using BERT's native WordPiece tokenizer, truncating sequences to 128 tokens, and converting them into PyTorch tensors. A baseline model was first implemented with fixed hyperparameters: a learning rate of 5e-5, batch size of 8, and three training epochs, using only a train–test split. Improvements included introducing a development set, early stopping, and Optuna-based hyperparameter tuning, optimizing learning rate, batch size, epochs, and weight decay to maximize the F1 score. These refinements shifted training from a fixed setup to a more adaptive, data-driven process, enhancing BERT's ability to model sarcasm's complex linguistic patterns.

BERT Variations

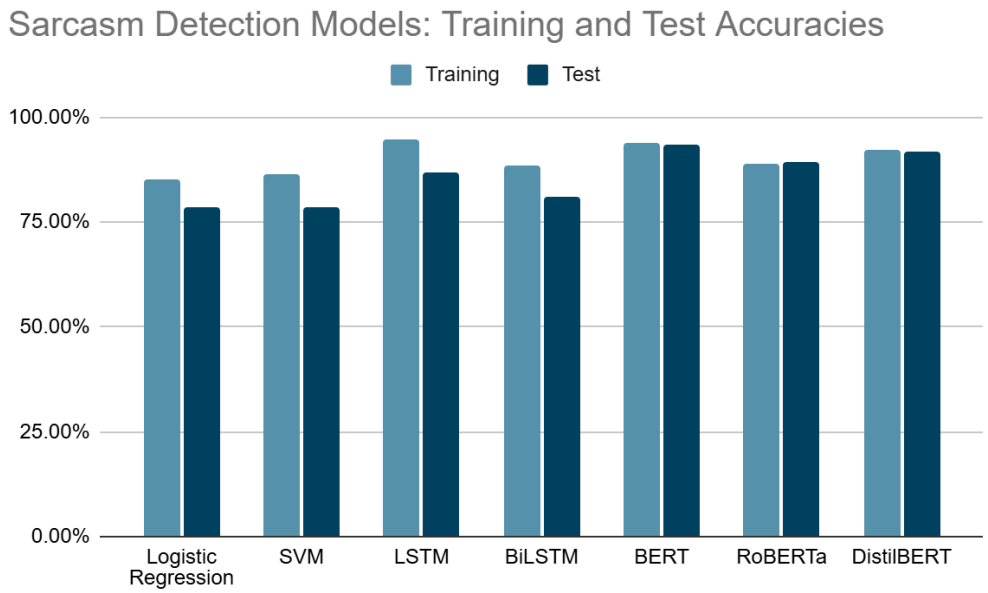
Some BERT variations like RoBERTa and DistilBERT were tested. Their performance was akin to BERT and thus they weren't explored more than the preliminary validation phase. The focus was centered on BERT as our primary model.

Evaluation

The various models were mainly evaluated on their training and test accuracy. The discrepancies in the training and testing accuracy showed which of the models were potentially overfitting the data. Additionally, the models were evaluated on their precision, recall and F1 score. A confusion matrix was also constructed for each model which further highlighted the overfitting.

Our dataset was labeled so all the machine learning tasks were supervised. The models were compared with each other to determine which performed the best.

Results



Traditional Machine Learning Models (SVM+Logistic Regression)

Both Logistic Regression and SVM exhibited overfitting, with training accuracies of 85.01% and 86.37%, but lower test accuracies of 78.58% and 78.76%, respectively. Despite its margin-based approach, SVM showed no significant improvement over Logistic Regression in generalization. Both models relied heavily on TF-IDF n-grams, capturing only surface-level lexical features, which limited their ability to handle the contextual nuances of sarcasm.

◆ SVM Results:
Training Accuracy: 0.8637
Test Accuracy: 0.7876

Test Set Classification Report:

	precision	recall	f1-score	support
0	0.79	0.84	0.82	2991
1	0.78	0.72	0.75	2330
accuracy			0.79	5321
macro avg	0.79	0.78	0.78	5321
weighted avg	0.79	0.79	0.79	5321

◆ Logistic Regression Results:
Training Accuracy: 0.8501
Test Accuracy: 0.7858

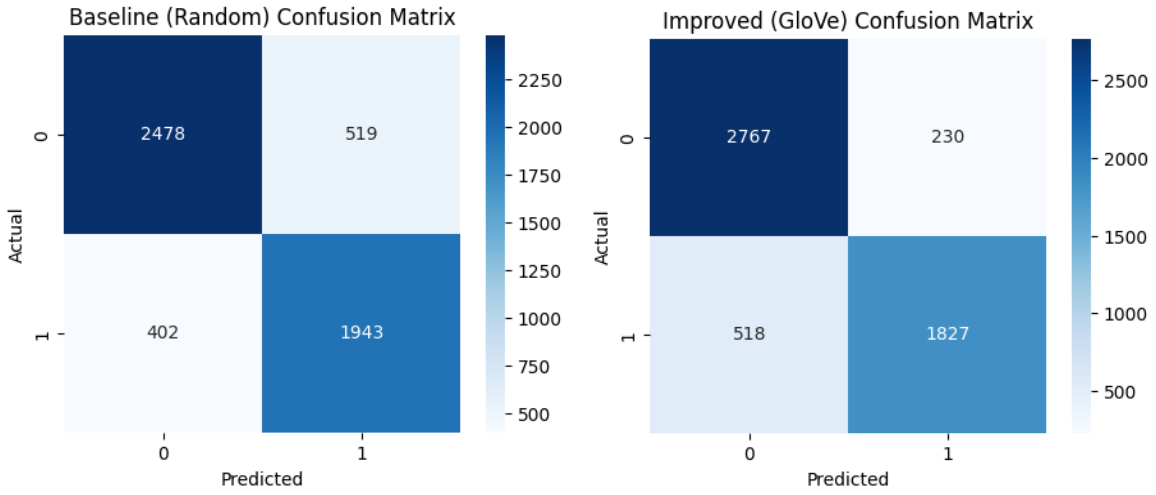
Test Set Classification Report:

	precision	recall	f1-score	support
0	0.78	0.85	0.82	2991
1	0.79	0.70	0.74	2330
accuracy			0.79	5321
macro avg	0.79	0.78	0.78	5321
weighted avg	0.79	0.79	0.78	5321

LSTM

On the held-out test set, the baseline LSTM achieved 0.8276 accuracy with precision 0.7892, recall 0.8286, and F1 0.8084. The improved (GloVe-initialized) model achieved 0.8600 accuracy with precision 0.8882, recall 0.7791, and F1 0.8301. Compared to the baseline, the improved model increased overall accuracy and substantially increased precision, indicating fewer false positives.

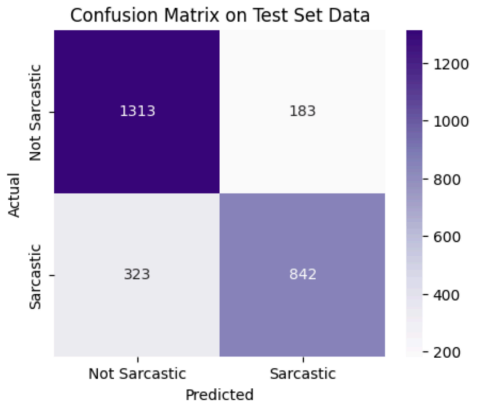
The confusion matrices reflect this tradeoff: the improved model reduced false positives from 519 → 230 (more non-sarcastic headlines correctly labeled), but increased false negatives from 402 → 518 (more sarcastic headlines missed). Overall, the GloVe initialization and regularization shifted the model toward being more conservative when predicting sarcasm.



BiLSTM

The initial model achieved a training accuracy of ~99% and a test accuracy of ~78%, indicating overfitting. The improved model, however, achieved a training accuracy of 94.74% and a test accuracy of 86.69%, indicating better generalization. The improvements, such as the use of pretrained GloVe embeddings, dropout, etc., contributed to vastly reducing overfitting and improved generalization. Below is the classification report and confusion matrix of the test set data on the improved model:

	precision	recall	f1-score	support
0	0.80	0.88	0.84	1496
1	0.82	0.72	0.77	1165
accuracy			0.81	2661
macro avg	0.81	0.80	0.80	2661
weighted avg	0.81	0.81	0.81	2661

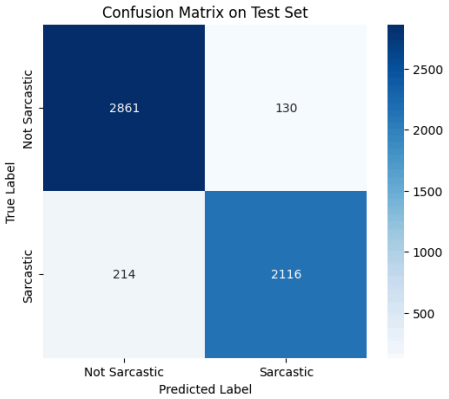


BERT

In the baseline system, which used only a train–test split and fixed hyperparameters, BERT achieved 93.18% accuracy and an F1 score of 92.21% on the test set, with precision and recall both slightly above 92%. The introduction of a development set, early stopping, and Optuna-based hyperparameter tuning improved generalization, increasing accuracy to 93.54% and F1 to 92.48%, primarily due to a precision boost from 92.26% to 94.21%, though recall dropped from 92.15% to 90.82%. These refinements—adjusting learning rate, batch size, epochs, and weight decay—transitioned training from a fixed setup to an adaptive optimization process, reducing overfitting. While recall slightly declined, the improved precision and overall stability

demonstrate the impact of targeted hyperparameter tuning. Below is the comparison of two versions of BERT and confusion matrix of the optimized BERT.

Metric	Baseline	Optimized	Improvement
Accuracy	93.18%	93.54%	0.36%
Precision	92.26%	94.21%	1.95%
Recall	92.15%	90.82%	-1.33%
F1 Score	92.21%	92.48%	0.27%



Outlook

We aimed to develop a sarcasm detector achieving at least 90% accuracy while exploring the subtleties of sequence classification, particularly how certain words or tokens might carry higher importance in detecting sarcastic intent. In the process, we learned about the constraints of a relatively small dataset leading to overfitting and observed how pretrained embeddings compare to training from scratch. Techniques like dropout and early stopping helped mitigate these issues, but the deeper challenge lies in determining whether the model genuinely detects sarcasm or merely associates certain tokens with sarcastic content.

Although our best-performing model (BERT) exceeded 90% detection accuracy, we remain uncertain if it captures the essence of sarcasm or simply relies on superficial cues. Moving forward, we would expand the dataset to include more diverse sources (e.g., Reddit) for broader coverage and explore large language models given sufficient computational resources. Testing on longer texts (beyond single headlines) would also shed light on context-driven cues for sarcasm. Overall, sarcasm detection remains both a challenging and rewarding frontier in deep learning.

References

Gregory, H., Li, S., Mohammadi, P., Tarn, N., Draelos, R., & Rudin, C. (2020a). A transformer approach to contextual sarcasm detection in Twitter. *Proceedings of the Second Workshop on Figurative Language Processing*. https://doi.org/10.18653/v1/2020.figlang-1.37

Kumar, Dr. M., & Patidar, A. (2021). Sarcasm detection using stacked bi-directional LSTM model. *3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 1–5. https://doi.org/10.1109/icac3n53548.2021.9725488

MirunaPislar. (n.d.). Mirunapislar/sarcasm-detection: Detecting sarcasm on Twitter using both traditional machine learning and deep learning techniques. GitHub. https://github.com/MirunaPislar/Sarcasm-Detection