Sarcasm–Aware Sentiment Classification

Le Zhang
University of Michigan
zhle@umich.edu

***Abstract*—This project explores how machine learning can be used to classify both sentiment and sarcasm in text. A multi-task BiLSTM model is trained on three public datasets and predicts both labels at the same time. After cleaning and padding the text, an attention pooling layer is used to help the model focus on important words. The model reaches high accuracy and balanced F1-scores on both tasks, showing it can handle sarcasm and emotion at the same time.**

I. Introduction

Sarcasm is common in everyday language, especially on social media. It often changes the meaning of a sentence and makes it harder for models to detect the actual emotion. Most sentiment models only look at surface words and miss the shift in meaning. So the goal here is to improve that by training one model to classify both sentiment and sarcasm together. To do this, I combine three public datasets from Hugging Face: the Tweet Sentiment Extraction dataset, the Sarcasm News Headline dataset, and the BESSTIE Reddit Sarcasm dataset.

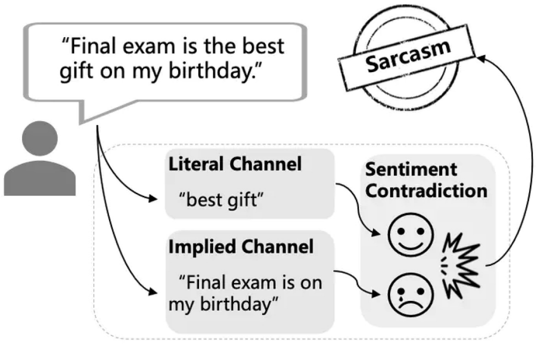
"Final exam is the best gift on my birthday."
Sarcasm
Literal Channel
"best gift"
Sentiment Contradiction
Implied Channel
"Final exam is on my birthday"

Fig. 1. Sarcasm Detection

The approach is inspired by recent work in multi-task learning. A paper by Tan et al. [1] shows that using one model to handle both sentiment and sarcasm helps improve accuracy and reduces repeated steps. Their model shares the same layers and splits the output, similar to the design used in this project. Another paper by Lakshmi [2] compares different deep learning methods for sarcasm detection, including CNN, LSTM, BiLSTM, and BERT. Their results show that BERT performs best in terms of accuracy, but BiLSTM is a simpler and faster option that still gives strong results, which makes it a practical choice for this study.

II. Method

To prepare the data, I first remove the neutral sentiment labels, as sarcasm is difficult to detect when the emotion is ambiguous. The sarcasm labels remain the same: 1 for sarcastic, 0 for not.

Each dataset is cleaned and changed to have the same three columns: text, sentiment, and sarcasm. Some texts are only labeled for one task. In that case, I set the missing label to –1 and use masking to skip it during training. This way, the model only updates weights when a valid label exists, and it can learn from both complete and incomplete data without confusion.

Then each sentence is cleaned by removing URLs, HTML tags, and common symbols like line breaks and HTML escape codes. I also convert all text to lowercase and replace common abbreviations like "idk," "u," and "btw" with their full forms. Most punctuation is removed, except apostrophes in words like "don't" or "I'm." These steps help reduce noise in the vocabulary and make similar phrases easier for the model to recognize.

After cleaning, the text is tokenized into words and converted into index sequences using a custom vocabulary. I remove rare words that appear only once to make the vocabulary smaller and more reliable. All sequences are padded to a fixed length of 100. This number is chosen because over 95% of the texts are shorter than 100 words, based on a histogram of word counts. Padding makes it possible to process the text in batches without losing important information from longer examples.

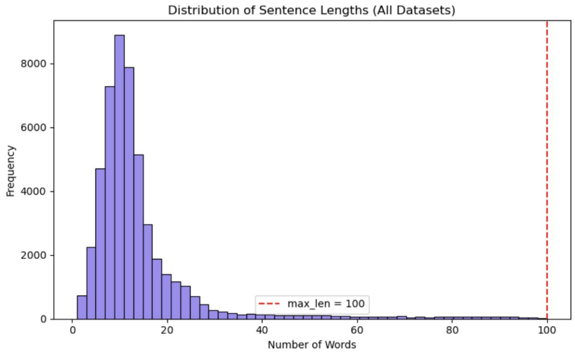

Fig. 2. Word count distributions for the three datasets.

The model is a multi-task BiLSTM. It starts with an embedding layer that turns each word index into a 128-dimensional vector. The sequence then goes into a two-layer bidirectional LSTM, which captures information from both directions of the sentence. To summarize the output, I use an attention pooling layer. This layer helps the model focus on the most useful words in the sentence, which is important for sarcasm since certain phrases have stronger meaning than others.

After pooling, a dropout layer is used to reduce overfitting. Then the model splits into two outputs. One layer predicts sentiment using softmax, and the other predicts sarcasm using a sigmoid function. Both outputs are trained at the same time, but each task uses its own loss. Cross-entropy loss is used for sentiment, and binary cross-entropy is used for sarcasm.

TABLE I
SAMPLE DISTRIBUTION

Subset	Number	Description
Total Samples	49,693	All examples from all datasets
Sentiment-labeled	19,497	Has sentiment labels (pos/neg)
Sarcasm-labeled	30,196	Has sarcasm labels (0 or 1)
Train Set	29,815	60% of the full dataset
Validation Set	9,939	20%, for tuning and early stopping
Test Set	9,939	20%, for final evaluation

The model is trained using Adam, which is a commonly used optimizer. The learning rate is set to 3e-4, and weight decay is 1e-5. I split the dataset into 60% training, 20% validation, and 20% testing. The batch size is 64. I use early stopping to stop training if the validation loss doesn't get better after two rounds. This helps avoid overfitting. A scheduler is also used to reduce the learning rate when the loss stops improving.

III. RESULT

The model is trained for 10 epochs with early stopping, and it stops at epoch 6. For sentiment, both training and validation losses drop quickly in the first few rounds. After that, the validation loss no longer improves. For sarcasm, the loss also decreases but more slowly. This is expected since sarcasm is harder to detect and often depends on phrasing or tone.

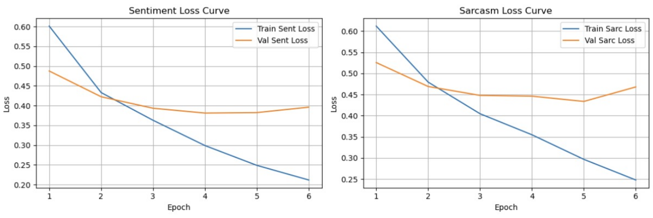

Fig. 3. Loss

By the end of training, the sentiment accuracy reaches 89.8% on the training set and 83.7% on the validation set. For sarcasm, the training accuracy is 88.3%, and the validation accuracy is 80.9%. On the test set, sentiment accuracy is 82.2% and sarcasm accuracy is 80.7%. These results are similar on all three sets, suggesting that the model generalizes well.

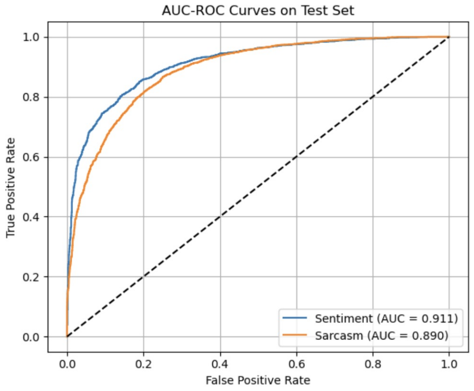

Fig. 4. AUC-ROC Curve

The model also performs well on other metrics. For sentiment, precision is 0.854, recall is 0.837, and the F1-score is 0.846. Most positive examples are correctly identified, and few are missed. For sarcasm, precision is 0.787 and recall is 0.821, leading to an F1-score of 0.803. This means the model catches most sarcastic examples, though it sometimes labels non-sarcastic ones as sarcastic. Nevertheless, the precision is still close to sentiment, so the predictions remain reliable.

AUC is 0.911 for sentiment and 0.890 for sarcasm. These scores show the model can clearly distinguish between the classes in both tasks. All performance metrics are summarized in Table.

TABLE II
TEST-SET PERFORMANCE

Task	Acc	Precision	Recall	F1	AUC
Sentiment	0.822	0.825	0.872	0.878	0.911
Sarcasm	0.807	0.798	0.801	0.795	0.890

Overall, the results show that the model can classify sentiment and sarcasm at the same time with high accuracy and balanced performance. While sarcasm is slightly more difficult, the difference across metrics is not large. This supports the idea that multi-task learning is effective in capturing both emotional tone and sarcastic intent.

IV. CONCLUSION

This project shows that both sentiment and sarcasm can be classified effectively using a single BiLSTM-based model. Masking lets the model learn even when only one label is present. Padding, attention pooling, and separate losses also help the model learn from structure, focus on important words, and balance both tasks. The results are consistent. In future work, pre-trained models like BERT could be tested for sarcasm that needs more context. [1]

REFERENCES

[1] Y. Y. Tan *et al.*, "Sentiment analysis and sarcasm detection using deep multi-task learning," *Applied Intelligence*, vol. 53, 2023.

[2] S. Lakshmi, "Sarcasm Detection Using Deep Learning," *Sentiment Analysis in NLP*, Elsevier, 2024.

[1] Code available at: https://github.com/LeZhang1117/stats507-coursework.git