Sarcasm Detection with TF–IDF + Linear SVM and BiLSTM + GloVe: A Comparative Study

Kelly Xu (nx27), Rija Khan (rk1047)

December 18, 2025

Abstract

This report presents a sarcasm detection system formulated as a binary text classification problem. We explore two main approaches: (1) a traditional machine learning pipeline based on word- and character-level TF–IDF features with a linear Support Vector Machine (SVM), and (2) a more advanced neural model: a bi-directional LSTM with pretrained GloVe embeddings and pooled sequence representations. The TF–IDF + SVM model achieves a strong macro F1-score around 0.88 on the held-out test set. The BiLSTM + GloVe model further improves generalization, reaching 88.92% test accuracy with macro F1 $\approx$ 0.888. Based on validation loss and overall test performance, we select the BiLSTM + GloVe model (`model.pth`) as the final model.

1 Introduction

Problem description

Sarcasm detection is an important and challenging task in natural language understanding. Sarcastic expressions often convey meanings that contrast with their literal interpretations, making them hard to identify. Robust sarcasm detection is critical for downstream applications such as sentiment analysis, opinion mining, and social media monitoring, where sarcastic content can significantly distort the overall sentiment.

In this project, we treat sarcasm detection as a supervised binary classification problem. Given an input sentence, the goal is to predict whether it is *sarcastic* (label 1) or *non-sarcastic* (label 0).

Approach overview

We investigate two different modeling approaches:

- A **TF–IDF + Linear SVM** model, where we extract both word-level and character-level TF–IDF features and train a linear SVM classifier.
- A **BiLSTM + GloVe** model implemented in PyTorch, which uses pretrained GloVe embeddings and a bidirectional LSTM encoder with pooled representations for classification.

We first performed data exploration and light preprocessing, then engineered features and designed the model architectures. For the TF–IDF + SVM model, we performed hyperparameter tuning via grid search with cross-validation. For the BiLSTM + GloVe model, we trained with early stopping based on validation loss (and track validation accuracy), and evaluated on the same held-out test set. We ultimately chose the BiLSTM + GloVe model checkpoint with the lowest validation loss (`model.pth`) as our final model.

Team member contributions

The project was completed collaboratively:

- Kelly: implemented **TF–IDF + LinearSVC model base**
- Rija: implemented **BiLSTM + GloVe model base**
- Both contributors jointly optimized the final BiLSTM + GloVe model, including preprocessing decisions, architecture refinements, hyperparameter tuning, training strategy (freeze/unfreeze embeddings, early stopping), and evaluation analysis.

2 Data Exploration and Preprocessing

2.1 Dataset statistics and characteristics

The datasets provided were these three CSV files:

- `train.csv` — training set,
- `valid.csv` — validation set,
- `test.csv` — held-out test set.

Each file contains the following two columns:

- `text`: the input sentence or headline,
- `label`: the ground-truth/label class (0 = non-sarcastic, 1 = sarcastic).

Using `train_df.info()` and `value_counts()`, we verified that all examples have valid labels and no missing text entries. The label distribution in the test set is:

- Class 0 (non-sarcastic): 526 examples,
- Class 1 (sarcastic): 440 examples.

Thus, the dataset is slightly imbalanced, motivating the use of macro F1-score as an important metric (in addition to accuracy).

2.2 Text length analysis

We computed the character-level length of each training example:

```
train_df["text_length"] = train_df["text"].astype(str).apply(len)
```

Descriptive statistics from `describe()` show that the average text length is on the order of a few tens of characters, with some very short headlines and a small number of longer ones. Inspecting the longest few examples did not reveal corrupted data such as HTML fragments (e.g.: "`<div>`") or repeated noise, so all examples were retained.

2.3 Preprocessing steps

We apply light preprocessing to avoid discarding useful lexical or stylistic cues:

- **Lowercasing**: For the word-level TF–IDF features, we set `lowercase=True` in the vectorizer. For the neural model, we lowercase tokens during tokenization.

- **Stopwords**: We do *not* remove stopwords. Function words and fixed expressions (e.g., "yeah right", "of course") can be crucial signals in sarcasm.
- **Stemming/Lemmatization**: We do not apply stemming or lemmatization, as we want to preserve surface forms that may carry stylistic information.

For the BiLSTM + GloVe model, we tokenize each sentence using the NLTK `word_tokenize`, map tokens to indices using a vocabulary built from the training set, and apply padding or truncation to a fixed maximum sequence length. To accommodate for occasionally very long headlines observed in the dataset, we use a relatively large maximum length (530 tokens) with `<PAD>` and `<UNK>` tokens.

2.4 Rationale for preprocessing choices

Sarcasm often relies on nuanced word choices, punctuation, and informal language. Aggressive text normalization can remove these cues. Therefore, we keep preprocessing minimal and rely on feature engineering (e.g., character n-grams in TF–IDF) and pretrained semantic representations (GloVe) to capture useful signals.

2.5 Data augmentation

We do not perform any data augmentation in this project. All experiments were conducted on the original training, validation, and test splits.

3 Feature Engineering

3.1 Feature extraction methods

We design two main types of features in our Linear SVM model:

(a) **Word-level TF–IDF**: extracted using `TfidfVectorizer` with word tokens. We consider both unigrams and bigrams and tune frequency thresholds.
(b) **Character-level TF–IDF**: extracted using `TfidfVectorizer` with `analyzer="char"`. We consider character n-grams of various lengths to capture spelling patterns, punctuation, and stylistic cues.

These two feature sets are combined using `FeatureUnion` to form a single high-dimensional sparse representation.

For the neural model, the primary "features" are **pretrained GloVe embeddings** (50-dimensional), which provide semantically meaningful vector representations for tokens.

3.2 Justification for feature choices

Word-level TF–IDF. Word-level features are effective for capturing semantic content and short phrases. In sarcasm detection, bigrams such as "yeah right", "as if", or "of course" are often strong indicators of sarcasm.

Character-level TF–IDF. Character n-grams are particularly useful for modeling:

- elongated words (e.g., "soooo good"),
- repeated punctuation (e.g., "!!!", "???"),
- informal spellings and stylized language.

Such patterns are common in sarcastic text and might not be captured well by pure word-level features.

Pretrained GloVe embeddings. GloVe vectors encode distributional semantics learned from large corpora, improving generalization beyond what can be learned from the project dataset alone. This is especially helpful for sarcasm detection, where subtle semantic relationships and context can matter.
We used this over Word2Vec since it limits a word's meaning to a small context window it lives in, which is not helpful in the case of sarcasm where the meaning of words depends more on a larger corpus of text (sentences, paragraphs, etc) that it lives in. In our experiments, training with Word2Vec embeddings showed that there was overfitting in the LSTM model due to increasing validation loss and decreasing validation accuracy after each iteration of the training loop.

3.3 Feature selection techniques

We did not employ an explicit feature selection step (e.g., chi-square filtering or L1-based selection). Linear SVMs are known to work well with high-dimensional sparse features, and in practice we observed no severe overfitting due to feature dimensionality.

4 Model Architecture and Selection

4.1 Models considered and why

We considered two models:

- **TF–IDF + Linear SVM**: a strong traditional baseline for text classification, efficient in high dimensions and robust under moderate data sizes.
- **BiLSTM + GloVe**: a neural network model that can capture word order and long-range dependencies, while leveraging pretrained embeddings for stronger semantic representations.
- **CNNs (not implemented)**: we did consider CNNs for this since it could potentially use filters to predict sarcasm in the text and is also used for text classification tasks. However, we chose to use LSTMs over CNNs due to the reasons explained above.

The comparison between the implemented models helps us understand whether a more advanced neural model can outperform a simpler linear model with engineered features on the given dataset.

4.2 TF–IDF + Linear SVM architecture

We implement the main model as a `sklearn` `Pipeline`:

```
baseline_char = Pipeline([
    ("vect", feature_union),  # word + char TF-IDF
    ("clf", LinearSVC())
])
```

where `feature_union` combines word-level and character-level TF–IDF vectorizers.

Hyperparameter choices and tuning process

We use `GridSearchCV` with 3-fold cross-validation to tune:

- `vect__word__ngram_range` $\in \{(1,1),(1,2)\}$,
- `vect__word__min_df` $\in \{1,2\}$,
- `vect__char__ngram_range` $\in \{(3,5),(4,6)\}$,
- `vect__char__min_df` $\in \{1,2\}$,
- `clf__C` $\in \{1,3,5\}$.

We optimize macro F1-score (`scoring="f1_macro"`) to treat both classes equally under the slight label imbalance. The best configuration is then used to retrain the model on the combined training and validation data.

Why this model suits the task

TF–IDF + Linear SVM is well-suited for sarcasm detection on this dataset because:

- It leverages rich lexical features (both word-level and character-level).
- Linear SVMs handle large sparse feature spaces effectively.
- The data size is moderate, which aligns well with the strengths of traditional linear models.

4.3 BiLSTM + GloVe architecture (final neural model)

The advanced LSTM model is implemented in PyTorch and consists of:

- An embedding layer initialized with **pretrained GloVe 50d** vectors.
- A **2-layer bidirectional LSTM** with hidden size 64 and dropout inside the LSTM.
- **Average + Max pooling** gives context of the overall sentence and emphasizes important features over the LSTM outputs. This helped avoid overfitting and reached better accuracy in testing.
- A dropout layer (0.2) on the pooled representation.
- A linear output layer producing a single logit for binary classification.

Embedding freezing/unfreezing. To stabilize early training and reduce overfitting, we freeze the embedding weights initially and unfreeze them after a few epochs (unfreeze at epoch 5 in our training loop).

Hyperparameters and selection

The BiLSTM + GloVe model is trained using:

- **Loss**: `BCEWithLogitsLoss`,
- **Optimizer**: Adam with learning rate 10^{-3} since it is a fast and efficient optimizer,
- **Batch size**: 64,
- **Max epochs**: 30 with early stopping (training stops once validation loss does not improve within 7 epochs),
- **Gradient clipping**: max norm 1.0 to avoid exploding gradients,
- **Decision rule**: $\sigma(\text{logit}) > 0.5$.

We save two checkpoints during training:

- `model.pth`: best (lowest) validation loss,
- `model_acc.pth`: best validation accuracy.

We select `model.pth` as the final model, based on its lowest validation loss and its best overall test performance and avoids overfitting compared to using best validation accuracy.

Why this model suits the task

This model suits sarcasm detection because:

- Pretrained GloVe embeddings improve semantic coverage and generalization.
- Bidirectionality captures context from both left-to-right and right-to-left.
- Avg+Max pooling summarizes sequence-level evidence robustly, helping with variable-length inputs.
- Regularization (dropout, early stopping, gradient clipping) improves stability and reduces overfitting.

5 Training Methodology

5.1 Training procedure and optimization

For the TF–IDF + SVM model:

1. Perform grid search with 3-fold cross-validation on the training set.
2. Select the model with the highest validation macro F1-score.
3. Retrain this best model on the union of the training and validation sets.
4. Evaluate the final model on the held-out test set.

For the BiLSTM + GloVe model:

1. Convert data to embeddings using pre-trained GloVe embeddings.
2. Train LSTM model on the training set using mini-batch gradient descent.
3. Monitor validation loss and validation accuracy after each epoch.
4. Save the best checkpoints for validation loss and validation accuracy.
5. Apply early stopping with patience of 7 epochs without validation-loss improvement.
6. Evaluate the final selected checkpoint (`model.pth`) on the test set.

5.2 Loss function and evaluation metrics

- **TF–IDF + SVM**: the `LinearSVC` classifier uses a hinge loss with L2 regularization.
- **BiLSTM + GloVe**: trained with `BCEWithLogitsLoss`.

Evaluation metrics include:

- Accuracy,
- Precision, recall, and F1-score for each class,
- Macro-averaged F1-score,
- Weighted-averaged F1-score,
- Confusion matrix.

5.3 Validation strategy

For TF–IDF + SVM, we use 3-fold cross-validation on the training set during hyperparameter tuning. For the BiLSTM + GloVe model, we use the provided held-out validation set (`valid.csv`) and early stopping based on validation loss.

5.4 Regularization and learning techniques

- In the SVM, the regularization strength is controlled by the parameter C, tuned via grid search.
- In the BiLSTM + GloVe model, we apply dropout, embedding freezing/unfreezing, gradient clipping, and early stopping to prevent overfitting and improve stability of the model.

6 Experiments and Results

6.1 TF–IDF + Linear SVM: baseline and final results

The TF–IDF + SVM model with both word-level and character-level features is our main traditional model. After tuning and retraining on the combined training and validation sets, it achieves the following performance on the test set (966 examples):

Class	Precision	Recall	F1-score	Support
0 (non-sarcastic)	0.88	0.90	0.89	526
1 (sarcastic)	0.88	0.85	0.87	440
Macro avg	0.88	0.88	0.88	966
Weighted avg	0.88	0.88	0.88	966

Table 1: Classification report for the TF–IDF + Linear SVM model on the test set.

The overall accuracy is 0.88, and both macro and weighted F1-scores are 0.88, indicating balanced performance across classes.

The confusion matrix is:

$$\begin{bmatrix} 474 & 52 \\ 64 & 376 \end{bmatrix}.$$

6.2 BiLSTM + GloVe results (final selected model)

We evaluate two models:

- `model.pth`: best validation loss (final selected and named as `model_weights.pth` in models folder),
- `model_acc.pth`: best validation accuracy.

We also stored the trained vocabulary and the embeddings in `vocab.pkl` and `embedding_matrix.pt` respectively for `predict_sarcasm.py`.

Final model (`model_weights.pth`) on the test set (966 examples). Testing this model on test.csv achieves:

- **Accuracy**: 88.92%,

Class	Precision	Recall	F1-score	Support
0 (non-sarcastic)	0.8844	0.9163	0.9001	526
1 (sarcastic)	0.8955	0.8568	0.8757	440
Macro avg	0.8899	0.8866	0.8879	966
Weighted avg	0.8895	0.8892	0.8890	966

Table 2: Test performance of the BiLSTM + GloVe model (`model.pth`).

- **Macro F1-score**: 0.8879.

The confusion matrix for `model.pth` is:

$$\begin{bmatrix} 482 & 44 \\ 63 & 377 \end{bmatrix},$$

which corresponds to 482 true negatives, 377 true positives, 44 false positives, and 63 false negatives.

Best-validation-accuracy checkpoint (`model_acc.pth`). For reference, `model_acc.pth` achieves 88.10% test accuracy, slightly worse than `model.pth` overall. Therefore, we select `model.pth` as the final model.

Note: Due to random weight initialization in the model, rerunning the training and test again may provide slightly different results, but not drastic enough for the model to behave differently.

predict_sarcasm.py Using the word embedding weights, vocabulary weights, and stored in their respective files, we loaded them in order to tokenize and create embeddings on the input CSV file given.

6.3 Ablation studies and component impact (qualitative)

Although we did not conduct a full numerical ablation study, several qualitative observations can be made:

- Adding character-level TF–IDF features on top of word-level TF–IDF improves robustness to elongated words, punctuation patterns, and informal spelling.
- For the neural model, pretrained GloVe embeddings and bidirectionality substantially improve performance compared to learning embeddings entirely from scratch.
- Using both average pooling and max pooling on the hidden states of the BiLSTM provides a more robust summary than using either alone. While this is an idea explored in CNN-based models (Doğan et al., 2020), we adapted this idea to BiLSTMs to turn the varied-length sequences of text into fixed-length inputs. Average pooling provides a more generalized representation of the input. However it struggles to distinguish important textual features (such as sarcasm in this project). This is where max pooling is useful, as it helps determine the important/strong features in the text to better predict/detect sarcasm. Using both poolings together showed slightly better accuracy in the model's testing. We observed that using only average pooling gave a slightly lower performance compared to max pooling and average+max pooling, which gave similar performances but average+max pooling was slightly higher.
- Regularization strategies (dropout, early stopping, gradient clipping, and freezing/unfreezing embeddings) improve training stability and generalization.

6.4 Confusion matrix and error analysis

Comparing the confusion matrices:

- The BiLSTM + GloVe model yields fewer false positives (44) than the TF–IDF + SVM model (52), and a comparable number of false negatives (63 vs. 64).
- Many misclassifications involve short, context-dependent sentences where sarcasm is ambiguous even for humans.
- Some sarcastic examples reference real-world events or require external knowledge, which neither model explicitly captures.

7 Discussion

7.1 What worked and what did not

The TF–IDF + Linear SVM model worked remarkably well, achieving strong macro F1 on the test set. Combining word- and character-level features was particularly effective. The grid search over hyperparameters allowed us to find a strong configuration without overfitting.

The BiLSTM + GloVe model further improved overall performance and matched or slightly exceeded the TF–IDF baseline on the test set. The largest benefits came from using pretrained embeddings, bidirectionality, and pooling-based sequence summarization, together with a stable training strategy (freeze/unfreeze embeddings, dropout, gradient clipping, and early stopping).

7.2 Insights about sarcasm detection

The experiments suggest that:

- Surface patterns such as bigrams and character n-grams capture a significant portion of sarcastic signals (TF–IDF + SVM).
- Pretrained semantic representations help capture meaning beyond surface lexical cues (BiLSTM + GloVe).
- However, some forms of sarcasm rely heavily on conversational context or world knowledge, which remains difficult for both approaches without external context modeling.

7.3 Model limitations

Both models have important limitations:

- Neither model uses conversational context or user history.
- The neural model uses a fixed maximum sequence length; very long inputs are truncated.
- No explicit external knowledge sources are incorporated.

7.4 Potential improvements

Future work could explore:

- Experimenting with pretrained transformer models (e.g., BERT, RoBERTa) fine-tuned on the sarcasm dataset.
- Incorporating conversational context or metadata.
- Adding features that explicitly model sentiment incongruity or contrast.
- Employing attention mechanisms on top of BiLSTM encoders.
- A denser BiLSTM could prove to be more useful in sarcasm detection.

8 Conclusion

In this project, we implemented and compared two approaches to sarcasm detection: a TF–IDF + Linear SVM model and an advanced BiLSTM + GloVe model. The TF–IDF baseline achieved strong performance with engineered lexical features. The BiLSTM + GloVe model achieved the best overall test performance (88.92% accuracy with macro F1 $\approx$ 0.888) and was selected as the final model (`model_weights.pth`) based on its lowest validation loss and strongest generalization.

8.1 Key takeaways:

- Lexical features (word + char TF–IDF) remain a strong baseline for sarcasm detection.
- Pretrained embeddings and bidirectional sequence modeling improve semantic generalization.
- Training stability techniques (dropout, gradient clipping, and early stopping) are important for neural models on moderate-sized datasets.

9 References

- Adam Optimizer: https://arxiv.org/abs/1412.6980
- Doğan, Yahya. "Which Pooling Method Is Better: Max, Avg, or Concat (Max, Avg)". Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering, vol. 66, no. 1, 2024, pp. 95-117, doi:10.33769/aupse.1356138.
- GloVe Documentation: https://nlp.stanford.edu/projects/glove/
- LinearSVC API: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
- LSTM Documentation: https://docs.pytorch.org/docs/stable/generated/torch.nn.LSTM.html
- PyTorch: https://pytorch.org/
- Scikit-learn: https://scikit-learn.org/
- TF–IDF feature extraction: https://scikit-learn.org/stable/modules/feature_extraction.html