Twitter Sentiment Analysis Using Classical & Neural Models

1. Problem Statement

Sentiment analysis on social media data is a crucial task for understanding public opinion and user behavior. This project aims to classify tweets from the Kaggle "Twitter Sentiment Dataset" into negative (-1), neutral (0), and positive (+1) sentiments. The challenge lies in handling large-scale textual data efficiently while comparing classical machine learning approaches with a neural network model (CNN–LSTM).

2. Research Questions

1. Which model achieves the best overall and per-class performance for predicting sentiment?
2. How do TF-IDF and Word2Vec features compare when used with classical models?
3. Does a CNN–LSTM hybrid outperform classical ensembles (Voting Classifier) on this dataset?

3. Dataset

- **Source:** Kaggle — Twitter Sentiment Dataset (Saurabh Shahane, 2021)
- **Columns:** clean_text (string), category (int: -1, 0, 1)
- **Size:** 162,980 tweets

4. Methods

1. **Exploratory Data Analysis (EDA) & Preprocessing**
    a. Class distribution, text length analysis, missing data handling
    b. Tokenization and stopword removal
    c. Sampling: due to memory constraints, we experimented with different sample sizes (1,000; 15,000; 20,000) to train models effectively
2. **Feature Engineering**
    a. **TF-IDF Features:** Classical baseline models
    b. **Word2Vec Features:** Average tweet embeddings for classical models
    c. **Neural Network Features:** Keras tokenizer with padding, feeding into CNN–LSTM
3. **Models**
    a. Classical: Decision Tree, KNN, Logistic Regression
    b. Ensemble: Voting Classifier combining best-performing classical models
    c. Neural: CNN–LSTM hybrid (embedding → convolution → LSTM → dense layers)
    d. **Best performing model:** CNN–LSTM
4. **Evaluation**
    a. Metrics: accuracy, precision, recall, macro F1, per-class F1

5. Work Organization

- **Joaquin:** EDA & preprocessing, main pipeline, CNN–LSTM implementation
- **Emin:** TF-IDF feature engineering
- **Will:** Word2Vec feature engineering

Workflow: At the start, the team defined a clear structure and split tasks. Each member worked separately due to different schedules and other project commitments. Weekly meetings during the Data Science lecture were used to check progress, coordinate next steps, and resolve issues.

6. Reflexion

From my perspective, the TF-IDF part of the project was mainly about understanding how much the feature representation alone can influence model quality. I was responsible for designing and implementing the TF-IDF pipeline and then evaluating different classical models on top of it. A practical challenge was choosing sensible hyperparameters for the vectorizer, because TF-IDF easily creates very high dimensional and sparse matrices. I experimented with the maximum number of features and n-gram ranges and had to balance model performance against training time and memory usage.

Working with several models on the same TF-IDF representation also showed me how differently algorithms react to sparse data. Logistic Regression and the VotingClassifier converged quickly and produced stable results, while KNN was slow and performed poorly. Seeing the confusion matrices and metrics side by side helped me understand that some algorithms are simply not well suited for high dimensional sparse text features. Overall, this part of the work strengthened my intuition for feature engineering and for how important it is to build a strong baseline before going to more complex neural models.

7. Conclusion and Future Work

Within the classical modelling track, TF-IDF with Logistic Regression turned out to be a very strong baseline. It achieved an accuracy and macro F1 around 0.83, which is close to the CNN-LSTM model and clearly better than all other classical models on both TF-IDF and Word2Vec. The VotingClassifier performed slightly worse, and KNN with TF-IDF was by far the weakest model with accuracy around 0.39. This confirms that a well tuned linear model on TF-IDF features can already capture most of the useful signal in short tweets, while more distance based methods struggle with the curse of dimensionality.

For future work, I would like to extend the TF-IDF experiments in three directions. First, testing character n-grams and sublinear term frequency scaling could further improve robustness to spelling variations and informal language. Second, applying dimensionality reduction methods such as Truncated SVD might reduce sparsity and make more complex models like KNN or ensemble methods more competitive. Third, it would be interesting to combine TF-IDF features with neural representations, for example by stacking a simple linear classifier on top of embeddings from a transformer model. This could keep the interpretability and efficiency of TF-IDF while benefiting from the contextual information captured by modern deep learning approaches.