SocialSense: Decoding Digital Emotions Across Meta, Instagram, and Twitter

Akshaya Thenkarai Lakshminarasimhan
athenkarailakshminar@ucsd.edu

1 Introduction

In today's digital age, social media platforms have become the primary channels for expressing emotions and sharing experiences. From joyful moments like "Enjoying a beautiful day at the park" to everyday neutral activities like "Trying out a new recipe," these digital expressions carry valuable emotional context. This project aims to harness the power of machine learning to analyze and classify sentiments across Facebook, Instagram, and Twitter, transforming casual social media interactions into meaningful insights that help bridge the gap between human emotion and digital understanding.

The primary objectives of this work include the following:

1. **Data Collection and Analysis** Upon searching for suitable datasets for social media sentiment analysis, I identified and selected a comprehensive dataset that includes data from the social media trinity (Facebook, Instagram, and Twitter). Examined the data to ensure that there are no data imbalances in terms of platform from which it has arrived.

2. **Data Preprocessing** This phase included cleaning and normalizing text content, removing characters such as emojis while preserving meaningful text data that contribute to sentiment understanding. The number of sentiment labels were 60 which were too many. So I had manually grouped relevant labels and mapped them to a single label that describes the entire group for the purpose of efficiency of the model. This resulted in 15 labels.

3. **Model Selection** Multiple modeling approaches were implemented to ensure robust sentiment classification. Traditional machine learning models like Naive Bayes and Logistic Regression established the baseline performance. Sequential LSTM models were employed to capture the temporal patterns in text data. Advanced BERT architectures were implemented to leverage contextual understanding, particularly important for social media's informal language patterns.

4. **Training Implementation** The training process involved strategically splitting the dataset into 80-10-10 training, validation and testing sets. Models were trained with carefully tuned hyperparameters to optimize performance. Early stopping mechanisms were implemented to prevent overfitting, particularly important given the diverse nature of social media text.

5. **Evaluation Methods** Comprehensive evaluation methods were employed to assess model performance. This included calculating accuracy metrics across different sentiment categories, implementing cross-validation to ensure robust performance assessment, monitoring model behavior across training epochs, and validating performance on unseen data to ensure generalization capability.

6. **Comparative Analysis** The final phase involved detailed comparison of model performances across different architectures. This included evaluating accuracy and efficiency metrics

2 Related Work

Sentiment analysis in social media has evolved significantly, with researchers exploring various

approaches to understand and classify user emotions across different platforms. Early work by Liu and Zhang (1) focused on basic text classification using lexicon-based methods, establishing foundational techniques for processing informal social media text and emoticons.

The field advanced with machine learning implementations, where Pak and Paroubek (2) demonstrated the effectiveness of Naive Bayes classifiers for Twitter sentiment analysis. Support Vector Machines (SVMs) gained prominence through the work of Go et al. (3), showing particular strength in handling informal language patterns and emoticons characteristic of social media posts like "Just finished an amazing workout! ".

A significant breakthrough came with deep learning approaches, as Wang et al. (4) demonstrated that LSTM networks could effectively capture long-term dependencies in social media text, outperforming traditional methods. The introduction of BERT by Devlin et al. (5) marked a revolutionary advancement, achieving state-of-the-art performance through its sophisticated contextual understanding of social media language patterns.

Recent research by Zhang and Wallace (6) has highlighted the potential of hybrid approaches that combine traditional machine learning with deep learning architectures. Their work emphasized the importance of specialized preprocessing techniques for social media text, particularly in handling platform-specific features like hashtags (Nature, Park), emojis, and varying text formats across different social media platforms (Twitter, Instagram, Facebook).

3 Dataset

3.1 Dataset Details and Key Statistics

The Social Media Sentiments Analysis Dataset comprises posts from three major platforms: Twitter, Instagram, and Facebook. Each entry contains rich metadata including text content, sentiment labels, timestamps, user information, hashtags, engagement metrics (retweets, likes), and geographical data. The dataset includes comprehensive information about social media interactions across different platforms and regions.

Key statistics from the dataset:

- **Temporal coverage:** 2023
- **Geographical distribution:** USA, Canada, UK, Australia

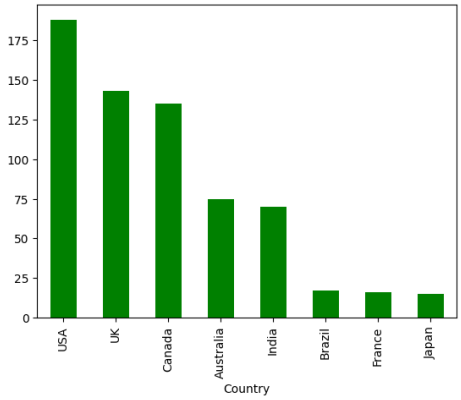

Figure 1: Geographic Distribution

- **Engagement metrics:** Retweets (5-20), Likes (10-40)
- **Platforms:** Twitter, Instagram, Facebook

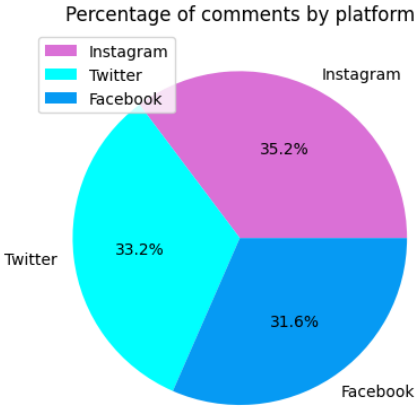

Figure 2: Data distribution across diverse platforms

Sample entries:

- **Sample 1:**

  Text: "Enjoying a beautiful day at the park!"
  Sentiment: Happy
  Platform: Twitter
  Hashtags: #Nature #Park

- **Sample 2:**

  Text: "Just finished an amazing workout!s"
  Sentiment: Happy
  Platform: Instagram
  Hashtags: #Fitness #Workout

3.2 Data Preprocessing

The preprocessing pipeline involved several crucial steps:

- Manually grouping the 60 sentiment labels to 15 labels which capture all the available labels and improvise the model performance during sentiment prediction.
- Removing tokens that do not contribute to the model.
- Text cleaning and handling of platform-specific features (hashtags, mentions)
- Filtered only essential features for the purpose of this task i.e text and sentiment.

3.3 Data Characteristics

The dataset presents several challenging aspects:

- Cross-platform content variations
- Diverse metadata requiring careful feature selection
- Multiple sentiment categories

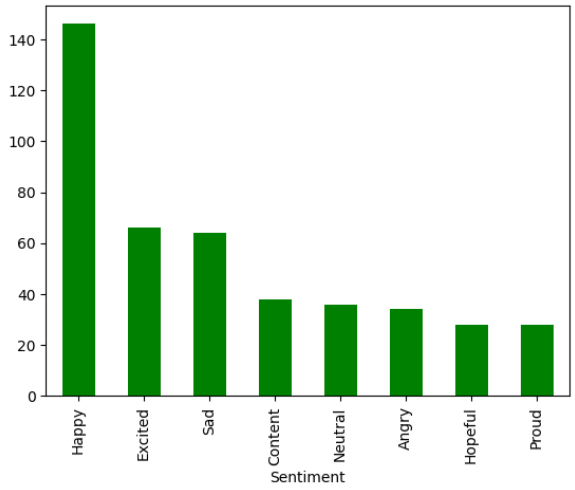

Figure 3: Sentiment Categories

- Regional and temporal variations in content
- Mixed content types including text, emojis, and hashtags

These characteristics make the dataset particularly suitable for multi-class sentiment analysis while presenting unique challenges in preprocessing and model development.

	Text	Sentiment
0	Enjoying a beautiful day at the park! ...	Happy
1	Traffic was terrible this morning. ...	Sad
2	Just finished an amazing workout! 💪 ...	Happy
3	Excited about the upcoming weekend getaway! ...	Happy
4	Trying out a new recipe for dinner tonight. ...	Neutral

Figure 4: Dataset Snapshot for model building

4 Baselines

This project implements and evaluates five different approaches for sentiment analysis on social media comments, ranging from traditional machine learning to advanced transformer architectures. The baseline models include Multinomial Naive Bayes and Logistic Regression, providing fundamental text classification capabilities. For mid-level complexity, two variants of Long Short-Term Memory (LSTM) networks are implemented and an enhanced version utilizing pre-trained GloVe embeddings. Finally, a BERT-based transformer model represents the advanced approach, leveraging contextual embeddings for improved sentiment understanding. This progression from simple to sophisticated architectures allows for comprehensive comparison of model performance across different levels of complexity.

4.1 Data Split and Model Validation

The dataset was split using a rigorous 80-10-10 approach to ensure unbiased evaluation of model performance:

- Training set (80%): 420 samples used for model training and hyperparameter tuning
- Validation set (10%): 53 samples used exclusively for evaluating model performance during development
- Test set (10%): 53 samples kept completely isolated until final evaluation

This strict separation ensures that:

- The test set remained completely unseen during both training and hyperparameter tuning
- The validation set was used solely for model evaluation during development
- No data leakage occurred between splits, providing unbiased performance estimates

All transformations and preprocessing steps were fit only on the training data and then applied to validation and test sets to prevent data leakage.

4.2 Models

1. **Multinomial Naive Bayes**

   Multinomial Naive Bayes classifier implemented with CountVectorizer for text feature extraction in a pipeline architecture. The model uses probabilistic classification based on Bayes' theorem with strong independence assumptions between features.

   Hyperparameters Tuned

   - CountVectorizer: max_df [0.5, 0.75, 1.0], ngram_range [(1,1), (1,2)]
   - MultinomialNB: alpha [0.1, 0.5, 1.0]

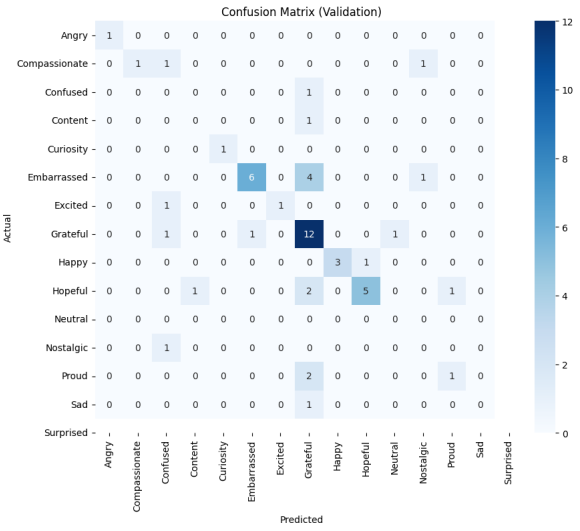

Figure 5: Confusion Matrix for Multinomial Naive Bayes

Performance

- Validation Accuracy: 0.58
- Test Accuracy: 0.55
- Best performing labels: Angry (1.00), Embarrassed (1.00)
- Poorest performing labels: Content (0.00), Curiosity (0.00)

2. **Logistic Regression** Logistic Regression is implemented as a linear classifier that predicts sentiment categories by applying a logistic function to a linear combination of input features. The model uses CountVectorizer for text feature extraction in a sequential process.

Hyperparameters Tuned

- max_iter: 1000 (Maximum number of iterations for solver convergence)
- solver: 'lbfgs' (default)
- multi_class: 'auto' (default)
- class_weight: None (uniform weights for all classes)

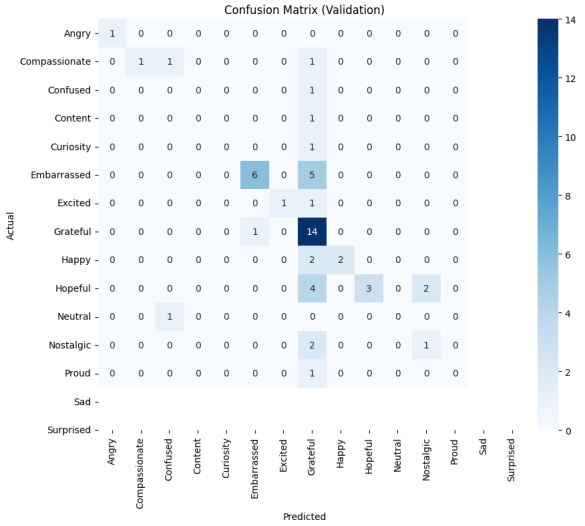

Figure 6: Confusion Matrix for Logistic Regression

Performance

- Validation Accuracy: 0.54
- Test Accuracy: 0.58
- Best performing labels: Angry (1.00), Compassionate (1.00)
- Poorest performing labels: Confused (0.00), Nostalgic (0.00)

3. **Long Short-Term Memory - Base Version**

   The model implements a sequential LSTM architecture with embedding layer and dropout for regularization. This represents a mid-level approach between traditional machine learning models and advanced transformers.

   Hyperparameters Tuned

   - Embedding dimension: [32, 64, 128]
   - Maximum sequence length: [10, 20, 50]
   - Batch size: [20, 30, 32]
   - Learning rate: [default Adam optimizer, SGD]
   - Epochs: [10, 20, 30]
   - Dropout rate: [0.2, 0.3]

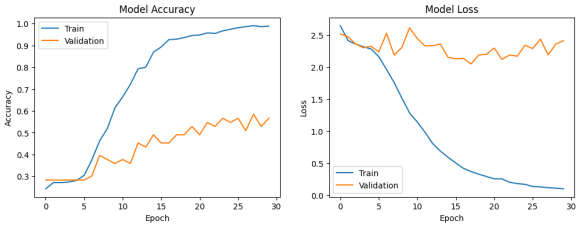

Figure 7: Accuracy/Loss graph LSTM Base version

Performance

- Training accuracy: 0.99 (final epoch)
- Validation accuracy: 0.58
- Test accuracy: 0.54

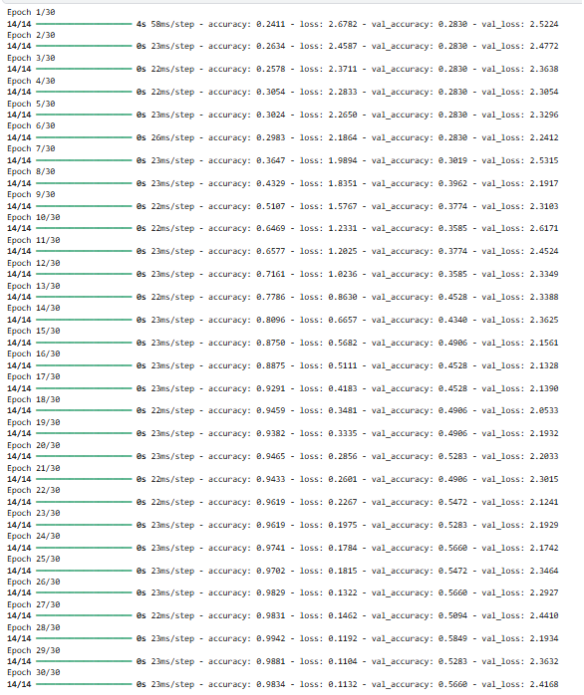

Figure 8: LSTM Base Version Training for 30 epochs

4. **Long Short-Term Memory - GloVe Embedding**

   This model implements a bidirectional LSTM architecture with pre-trained GloVe embeddings for enhanced text representation. The model leverages transfer learning by using pre-trained word embeddings while maintaining the ability to capture sequential patterns in the text data.

   Hyperparameters Tuned

   - max_words: [500, 1000, 10000] (vocabulary size)
   - max_length: [20, 30, 50] (sequence length)
   - embedding_dim: 100 (GloVe dimensions)
   - batch_size: 32
   - epochs: 30 with early stopping
   - dropout_rate: [0.2, 0.3]

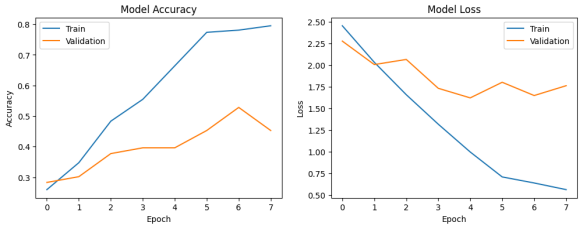

Figure 9: Accuracy/Loss graph LSTM with GloVe

Performance

- Training accuracy: 0.78 (final epoch)
- Validation accuracy: 0.53
- Test accuracy: 0.60

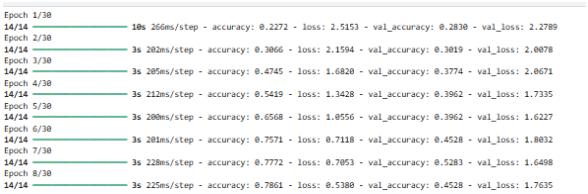

Figure 10: LSTM GloVe Training with Early Stopping

5. **BERT** The BERT model represents an advanced approach utilizing pre-trained contextual embeddings for sentiment classification. The model processes text bidirectionally through self-attention mechanisms to capture complex relationships between words.

   Hyperparameters Tuned

   - Pre-trained model: bert-base-uncased
   - Max sequence length: [64, 128, 256]
   - Batch size: [16, 32]
   - Learning rate: [1e-5, 2e-5]
   - Early stopping patience: 3

   Performance

   - Validation Accuracy: 0.774
   - Test Accuracy: 0.849
   - Best performing labels: Angry (1.00), Compassionate (1.00)

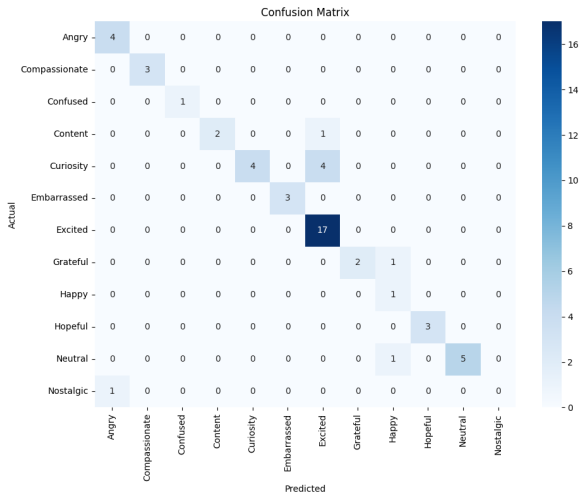

Figure 11: Confusion Matrix for BERT

- Poorest performing labels: Confused (0.00), Nostalgic (0.00)

```
Epoch 1: 100%|██████████| 27/27 [04:19<00:00,  9.60s/it, loss=2.139]
Epoch 1:
Average training loss: 2.644
Validation Accuracy: 0.283
Epoch 2: 100%|██████████| 27/27 [04:19<00:00,  9.62s/it, loss=2.505]
Epoch 2:
Average training loss: 2.239
Validation Accuracy: 0.283
Epoch 3: 100%|██████████| 27/27 [04:15<00:00,  9.48s/it, loss=1.477]
Epoch 3:
Average training loss: 1.921
Validation Accuracy: 0.396
Epoch 4: 100%|██████████| 27/27 [04:18<00:00,  9.57s/it, loss=1.226]
Epoch 4:
Average training loss: 1.618
Validation Accuracy: 0.415
Epoch 5: 100%|██████████| 27/27 [04:20<00:00,  9.64s/it, loss=0.979]
Epoch 5:
Average training loss: 1.279
Validation Accuracy: 0.566
Epoch 6: 100%|██████████| 27/27 [04:16<00:00,  9.49s/it, loss=0.665]
Epoch 6:
Average training loss: 0.962
Validation Accuracy: 0.660
Epoch 7: 100%|██████████| 27/27 [04:19<00:00,  9.59s/it, loss=0.544]
Epoch 7:
Average training loss: 0.708
Validation Accuracy: 0.717
Epoch 8: 100%|██████████| 27/27 [04:17<00:00,  9.52s/it, loss=0.161]
Epoch 8:
Average training loss: 0.468
Validation Accuracy: 0.755
Epoch 9: 100%|██████████| 27/27 [04:21<00:00,  9.68s/it, loss=0.113]
Epoch 9:
Average training loss: 0.306
Validation Accuracy: 0.774
Epoch 10: 100%|██████████| 27/27 [04:17<00:00,  9.55s/it, loss=0.071]
Epoch 10:
Average training loss: 0.198
Validation Accuracy: 0.774
Best validation accuracy: 0.774
```

Figure 12: BERT Fine-Tuning with 10 epochs

5 Approach

5.1 Conceptual Approach

This project implements a comprehensive sentiment analysis system using multiple models of increasing complexity. The approach progresses from traditional machine learning (Naive Bayes, Logistic Regression) through mid-level neural networks (LSTM with and without pre-trained embeddings) to advanced transformer architectures (BERT). Each model was chosen to explore different aspects of text classification, from simple bag-of-words to contextual embeddings.

5.2 Working Implementation

Successfully implemented five distinct models:

- Baseline Models: Multinomial Naive Bayes with CountVectorizer and Logistic Regression with CountVectorizer
- Neural Networks: Basic LSTM with tokenization and LSTM with GloVe embeddings
- Advanced Model: BERT with fine-tuning

All implementations are contained in jupyter notebook, each with complete preprocessing, training, and evaluation pipelines.

5.3 Compute Environment

The experiments were conducted on Kaggle's notebook environment as well as on Google Colab. Kaggle environment was best to leverage GPU runtime capabilities which provided more computational resources compared to alternatives like Google Colab. Google Colab crashes during training models which resulted in retraining them with different check points, loading weights, continue training for larger epochs, applying early-stopping criterions which was tedious, time consuming and delayed the process. The Specific implementation details include:

- GPU Runtime: Utilized Kaggle's T4 X 2 GPU for model training and evaluation
- Memory Management: Effectively handled large model training with Kaggle's 13GB GPU memory
- Session Stability: Kaggle's longer runtime sessions allowed uninterrupted training of complex models
- Uploaded glove.6B.100d.txt for LSTM variant 2 implementation.

Some Hacks: During the experiments, initially I began my work in Google Colab. But their compute resources were not sufficient for my experiments. I had to switch to the Kaggle's infrastructure which proved sufficient for training all models, including the resource-intensive BERT model without crashing. The platform's robust

GPU support and stable environment enabled efficient execution of all experiments without the need for significant optimization. One challenge was that the notebook did not get auto-saved with results and had to retrain the models to record the values into my report which consumed a significant amount of my time and delayed the process. A workaround that I incorporated is to save the model's best weight so that I can load it on my next run and continue the training and evaluation process. Also I've included early-stopping mechanisms wherever possible and save the best possible model weights to continue the training process.

5.4 Runtime

Model training times varied significantly:

- Baseline Models (Naive Bayes, Logistic Regression): 1 minute for a single run
- Basic LSTM: 120 minutes for hyperparameter tuning and best param detection
- LSTM with GloVe: 40 minutes with early stopping and best param detection
- BERT: 1.5 hours for 10 epochs, 15 hours fine tuning with different parameters

Model performance comparison:

Model	Val. Acc.	Test Acc.
Naive Bayes	0.58	0.55
Logistic Regression	0.5471	0.5849
Basic LSTM	0.58	0.5471
LSTM with GloVe	0.5283	0.6038
BERT	0.774	0.849

Table 1: Performance Comparison of Different Models

5.5 Additional Considerations

- BERT significantly outperformed other models on validation and test set
- GloVe embeddings improved LSTM performance over basic tokenization
- Class imbalance affected model performance across all architectures
- Early stopping helped prevent overfitting in neural network models

6 Error Analysis

Manual analysis of failed cases revealed several patterns:

Common Failure Cases:

1. **Mixed Emotions:**
   - Posts containing multiple sentiments were often misclassified
   - Example: "Excited about the weekend but sad it's raining"
2. **Platform-Specific Language:**
   - Different linguistic patterns across platforms
   - Instagram posts with heavy emoji usage posed challenges
   - Example: "Just finished an amazing workout! (emoji of arm)" (platform-specific context)
3. **Contextual Dependencies:**
   - Posts requiring broader context for interpretation
   - Example: "Here we go again" (ambiguous without context)
4. **Regional Variations:**
   - Geographical differences in expression patterns
   - Example: Different sentiment expressions across USA, UK, Australia

The BERT model showed better handling of these cases compared to LSTM and baseline models, particularly with contextual understanding and emoji interpretation.

7 Conclusion

7.1 Key Takeaways

The implementation of multiple models for social media sentiment analysis revealed several important insights. Traditional models like Naive Bayes and Logistic Regression provided strong baselines, while BERT demonstrated superior performance in handling contextual nuances of social media text. The cross-platform nature of the dataset (Twitter, Instagram, Facebook) highlighted the importance of robust pre-processing techniques and the challenge of handling platform-specific features.

7.2 Challenges Faced

Several aspects proved more challenging than anticipated:

- Managing computational resources for BERT training on Google Colab, eventually shipping the code to Kaggle enviroment and carrying out my experiments.
- Processing emojis and special characters while maintaining their sentiment value
- Handling platform-specific language patterns and informal text

7.3 Surprising Observations

One surprising finding was the relatively strong performance of traditional models despite their simplicity. The LSTM model achieved 60% accuracy, demonstrating that sequential models can effectively capture sentiment patterns in social media text. However, BERT's superior performance (84.9% accuracy) validated the importance of contextual understanding in sentiment analysis.

7.4 Future Directions

Future work could explore several promising directions:

- Implementing multi-modal analysis incorporating images and hashtags
- Developing platform-specific models to better handle unique linguistic patterns
- Exploring ensemble methods combining traditional and deep learning approaches
- Expanding the dataset to include more diverse languages and regional expressions
- Investigating zero-shot learning for emerging sentiment categories

These enhancements could further improve the model's ability to understand and classify the complex emotional expressions found in social media content.

8 Acknowledgements

Sections Written or Assisted by AI: The majority of this report was written by me, with assistance from ChatGPT for proof- reading and improving phrasing in certain sections.

References

[1] Liu, B., Zhang, L. (2012). A survey of opinion mining and sentiment analysis. In Mining text data (pp. 415-463).

[2] Pak, A., Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In LREc (Vol. 10, No. 2010, pp. 1320-1326).

[3] Go, A., Bhayani, R., Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N project report, Stanford, 1(12), 2009.

[4] Wang, X., Liu, Y., Sun, C. (2016). A deep learning approach to sentiment analysis of social media text. Neural Computing and Applications.

[5] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding.

[6] Zhang, Y., Wallace, B. (2017). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification.