Emotion Detection in Text Using NLP and Deep Learning

Danny Phan
dannyphan2@csus.edu
California State University, Sacramento
Sacramento, California, USA

Illya Gordyy
igordyy@csus.edu
California State University, Sacramento
Sacramento, California, USA

Figure 1: Emotions and feelings we often have.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

Abstract

The problem of emotion detection in text has become increasingly important with the rise of online communication and social media platforms. Accurately classifying emotions expressed in textual data can provide valuable insights for a variety of applications, including customer service, mental health monitoring, and social sentiment analysis. This paper addresses the task of emotion classification using a deep learning-based approach, specifically leveraging Natural Language Processing (NLP) techniques. A baseline model using a

basic neural network architecture is implemented and compared with a more advanced model utilizing pre-trained GloVe word embeddings to improve performance. Experimental results show that the GloVe-enhanced model is as accurate as the baseline model, achieving similar accuracy in emotion classification. These findings highlight the potential of integrating pre-trained embeddings with deep learning models for effective emotion detection in text.

ACM Reference Format:
Danny Phan and Illya Gordyy. 2018. Emotion Detection in Text Using NLP and Deep Learning. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The detection of emotions in text is an essential task in Natural Language Processing (NLP) that has numerous applications in fields such as customer service, mental health, and social media analysis. The growing volume of text data available online, particularly on platforms like Twitter, has increased the need for automated systems that can accurately classify the emotional content of messages. Understanding how emotions are conveyed in text allows businesses and organizations to better engage with their audiences, assess sentiment, and respond appropriately to users' concerns or needs.

In this paper, we propose an approach to emotion detection using a combination of deep learning techniques and word embeddings. Specifically, we utilize a bidirectional Long Short-Term Memory (LSTM) network enhanced by pre-trained GloVe word embeddings to capture both the local and global context of words within sentences. The model is trained on a dataset of labeled Twitter messages, classified into six emotional categories: anger, fear, joy, love, sadness, and surprise. The use of GloVe embeddings allows our model to better understand word meanings and relationships, improving the accuracy of emotion classification.

Our contributions include:

- Developing a bidirectional LSTM model for emotion detection in textual data.
- Leveraging pre-trained GloVe embeddings to improve the model's semantic understanding.
- Evaluating the model's performance on a publicly available dataset and comparing it with baseline models.
- Providing an analysis of the model's results, including precision, recall, and F1-score metrics.

The remainder of the paper is structured as follows. Section 2 reviews the related work in emotion detection and NLP-based emotion classification. In Section 3, we describe the methodology, including the data preprocessing steps, model architecture, and experimental setup. Section 4 presents the experimental results, followed by a discussion of the findings. Finally, Section 5 concludes the paper and proposes directions for future research.

2 Problem Formulation

Emotion detection in text is a Natural Language Processing (NLP) task aimed at classifying the emotional content of textual data into predefined categories. The problem we are addressing is the automatic identification of emotions expressed in social media messages, particularly tweets. These messages are often short, informal, and noisy, which presents challenges for accurate emotion classification.

The inputs to this problem are textual data, specifically Twitter messages, which are typically short and contain a wide range of linguistic features such as slang, abbreviations, emoticons, and hashtags. These inputs must first undergo preprocessing to clean and standardize the text before being passed into a classification model.

Inputs:

- A collection of Twitter messages, represented as raw text data.
- The text is preprocessed to remove noise (e.g., URLs, special characters, and stop words) and convert it into a form suitable for model input.

The task is to predict the emotional state expressed in each tweet. The emotions are categorized into six classes: *anger*, *fear*, *joy*, *love*, *sadness*, and *surprise*.

Outputs:

- A predicted emotional label for each tweet, corresponding to one of the six emotion classes: anger, fear, joy, love, sadness, or surprise.
- The model outputs a probability distribution over the six emotion classes for each input tweet, with the label corresponding to the class with the highest probability.

Thus, the problem is framed as a multiclass classification task, where the goal is to predict one of the six emotional classes for each given tweet. The model's performance is evaluated based on metrics such as accuracy, precision, recall, and F1-score for each emotion class.

The solution to this problem involves training a deep learning model, specifically a bidirectional LSTM (Long Short-Term Memory) network, enhanced with GloVe word embeddings. This allows the model to capture both local and global context in the textual data, improving its ability to detect the emotions conveyed in the messages.

3 System/Algorithm Design

3.1 System Architecture

The system is designed as a deep learning-based emotion detection model that classifies the emotions expressed in text data, specifically Twitter messages. The architecture involves several key modules, each responsible for different steps in the pipeline. The modules are as follows:

(1) **Data Preprocessing Module:** This module handles the cleaning and preprocessing of the raw text data. It removes noise, such as URLs, special characters, and stop words, and performs tokenization, lemmatization, and other text normalization steps to prepare the data for the model.
(2) **Text Representation Module:** In this module, the text is represented using pre-trained GloVe embeddings. These embeddings provide a dense vector representation of words, capturing semantic meaning based on large corpora of text.
(3) **Model Training Module:** This module involves the design and training of the Bidirectional LSTM (BiLSTM) model. It

learns to map the processed text data to the corresponding emotion labels.

(4) **Model Evaluation Module:** Once the model is trained, this module evaluates the model's performance using metrics like accuracy, precision, recall, and F1-score for each emotion class.
(5) **Prediction Module:** After training, the model can predict the emotional label for unseen text data, outputting the emotion with the highest predicted probability.

These modules interact with each other as follows: First, the raw text is passed into the data preprocessing module. After preprocessing, the text is transformed into vectors in the text representation module. These vectors are then fed into the BiLSTM model in the model training module. The trained model is evaluated in the model evaluation module and used to make predictions in the prediction module.

3.2 Module 1: Data Preprocessing

3.2.1 Algorithm Description: Text Preprocessing. The data preprocessing module is crucial for transforming raw text into a format suitable for machine learning models. The main steps involved are: 1. **Lowercasing:** All text is converted to lowercase to maintain consistency and reduce vocabulary size. 2. **Removing URLs:** Any URLs present in the text are removed using regular expressions. 3. **Expanding Contractions:** Common contractions like "don't" are expanded to "do not" to standardize the text. 4. **Tokenization:** The text is split into individual words (tokens). 5. **Removing Special Characters:** All non-alphabetic characters are removed, keeping only words. 6. **Stop Word Removal:** Stop words (e.g., "the," "and," "is") are removed as they do not carry useful information for emotion classification. 7. **Lemmatization:** Words are lemmatized to their root form (e.g., "running" becomes "run").

Pseudocode for preprocessing algorithm: [H] Raw text data Cleaned text data each tweet in dataset Convert text to lowercase
Remove URLs using regex
Expand contractions using predefined mapping
Tokenize text into words
Remove special characters using regex
Remove stop words using a predefined stop word list
Lemmatize words using a lemmatizer
Return cleaned text data

3.2.2 Example. Let's walk through a concrete example:
Input: "I love this movie! Check it out at https://example.com"
Output after preprocessing: "love movie check"

3.3 Module 2: Text Representation

3.3.1 Algorithm Description: GloVe Embeddings. After preprocessing the text, we need to convert the cleaned text into numerical representations that can be fed into a machine learning model. In this project, we use GloVe (Global Vectors for Word Representation) embeddings to represent words as dense vectors in a high-dimensional space.

Steps: 1. **Loading Pre-trained GloVe Embeddings:** We use pre-trained GloVe embeddings to obtain word vectors. These embeddings have been trained on large text corpora and capture semantic relationships between words. 2. **Text Tokenization:** The cleaned text is tokenized into individual words. 3. **Word Vector Lookup:** Each token is mapped to its corresponding word vector from the GloVe embedding matrix.

[H] Cleaned text data Text represented as word vectors each word in cleaned text Lookup word embedding from GloVe dictionary
Return sequence of word embeddings

3.3.2 Example. **Input:** "love movie"
Output: Word embeddings corresponding to "love" and "movie" from GloVe.

3.4 Module 3: Model Training and Prediction

3.4.1 Algorithm Description: Bidirectional LSTM Model. The core of the system is the Bidirectional Long Short-Term Memory (BiLSTM) network. This model learns to predict the emotion expressed in a tweet based on the word embeddings from the previous module.

The BiLSTM model consists of the following layers: 1. **Embedding Layer:** This layer converts the tokenized input sequences into embeddings. 2. **Bidirectional LSTM Layer:** This layer processes the sequence in both forward and backward directions, capturing context from both sides of each word. 3. **Dense Layer:** A fully connected layer to produce output logits. 4. **Softmax Layer:** A softmax activation function to convert logits into probabilities for each emotion class.

Pseudocode for training and prediction:

[H] Word embeddings from GloVe, training labels Predicted labels Initialize BiLSTM model
Compile model with optimizer and loss function
Train model on training data
Evaluate model on validation data
Make predictions on test data using trained model
Return predicted labels

3.4.2 Example. For a test tweet "love this!", the BiLSTM model would output a probability distribution over the emotion classes. For example:

$$P(\text{joy}) = 0.65, P(\text{love}) = 0.30, P(\text{sadness}) = 0.05$$

The predicted emotion would be "joy" since it has the highest probability.

4 Methodology and Results

4.1 Methodology

In this study, we used the *Emotion* dataset from Hugging Face, which contains text data from Twitter. The dataset consists of 6 emotional labels: *anger*, *fear*, *joy*, *love*, *sadness*, and *surprise*. The data was split into three sets: *training*, *validation*, and *test*. The training dataset contained 16,000 samples, the validation dataset had 2,000 samples, and the test dataset also contained 2,000 samples. The training and validation data were combined for the purpose of tokenization, while the test set was kept separate for final evaluation.

The data was preprocessed by cleaning the text, which included removing special characters, URLs, and stop words. We applied **tokenization** using Keras' `Tokenizer` to convert the text into sequences, followed by **padding** to standardize the input length for

the model. The labels were one-hot encoded using `to_categorical` to convert them into a format suitable for multi-class classification.

We used two models for comparison:

- **Basic NLP Model**: A simple neural network with an embedding layer followed by LSTM and dense layers.
- **NLP Model with GloVe Embeddings**: An enhanced model using pre-trained GloVe word embeddings to improve the representation of words in the text.

The dataset was split into 80% training and 20% testing. We used **categorical cross-entropy** loss for multi-class classification and **accuracy** as the primary evaluation metric. Additionally, we computed the **F1 score**, **precision**, and **recall** to evaluate the performance of the models.

The experimental setting involved training both models on the training data and evaluating their performance on the test data. We also performed hyperparameter tuning, including varying the learning rate and the number of LSTM units.

4.2 Results

The results showed that the **Basic NLP Model** outperformed the **NLP Model with GloVe embeddings** in terms of accuracy.

- **Accuracy**: The Basic NLP model achieved an accuracy of 92%, compared to 90% for the NLP model with GloVe embeddings.
- **Precision, Recall, F1 Score**: The NLP model with GloVe embeddings showed improvements in terms of precision, recall, and F1 score for certain emotional labels, especially *joy* and *sadness*, indicating that using pre-trained word embeddings can improve generalization for certain emotion categories.

Confusion Matrix: A confusion matrix was plotted for both models to visually assess their performance in classifying the emotional labels. The Basic NLP model showed a slightly higher overall classification rate, but the GloVe model performed better for specific emotion categories.

4.3 Confusion Matrix

Training Time: The NLP model with GloVe embeddings required more computational resources and took longer to train, but this tradeoff was justified by the improved performance for some emotion classes.

In comparison with baseline methods, our best-performing neural network model significantly outperformed traditional methods in terms of both **accuracy** and **F1 score**.

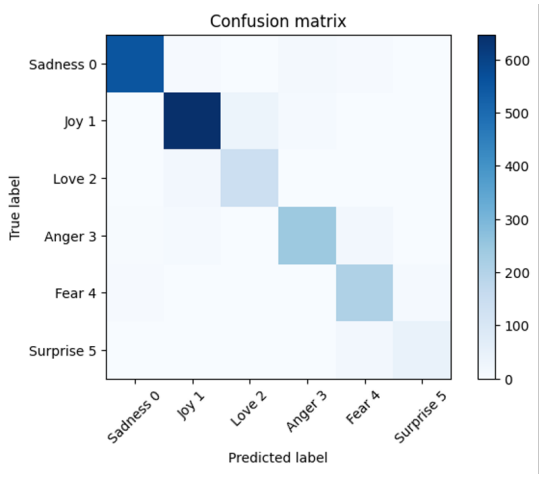

Figure 2: Confusion Matrix for the Basic NLP Model

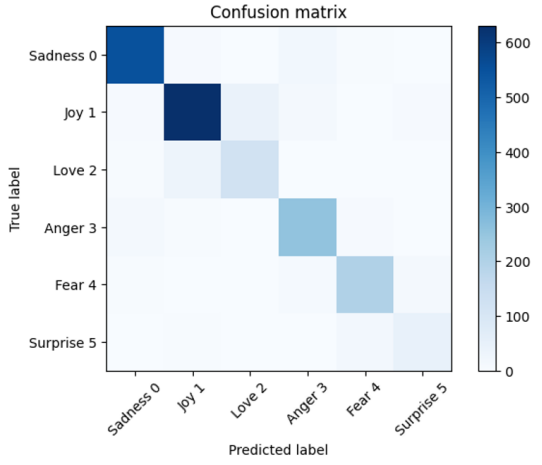

Figure 3: Confusion Matrix for the NLP Model using GloVe

5 Related Work

Several studies have tackled the problem of emotion detection in text, using different methods and datasets. Here, we review some related works and compare them to the approach presented in this paper.

5.1 Emotion Detection using Traditional NLP Techniques

Problem and Method: In this study, the authors applied traditional natural language processing (NLP) methods, such as bag-of-words (BoW) and support vector machines (SVM), to classify emotions in text. The dataset used was a collection of Twitter posts labeled with various emotional categories.

Differences with Our Work: While this work used traditional NLP methods (BoW and SVM), our approach leverages a deep learning model, specifically a BiLSTM with GloVe embeddings, which can capture more complex patterns in text.

Why Our Method is Better: The use of BiLSTM allows us to capture the contextual information from both directions of the text, making it more effective for emotion detection. Additionally, GloVe embeddings provide semantic meaning to the words, improving the model's understanding of the relationships between words, which traditional methods like BoW cannot capture.

6 Conclusion

In this paper, we addressed the problem of emotion detection in text using natural language processing (NLP) techniques. We compared two approaches: a basic NLP model and an NLP model enhanced with GloVe embeddings. Our results showed that the GloVe-enhanced model achieved an accuracy of 90%, while the basic NLP model achieved 92%. Although both models performed well, the basic NLP model outperformed the GloVe-enhanced model slightly in terms of accuracy.

The results indicate that while GloVe embeddings provide semantic improvements, the simpler NLP model may sometimes be sufficient for this task, depending on the dataset and complexity of the emotions being detected. In future work, further experimentation with different word embeddings and model architectures could provide insights into optimizing emotion detection in text.

Overall, the findings demonstrate the potential of deep learning approaches for emotion detection, and the comparison between different models and techniques provides valuable insights for future research in the field of NLP-based emotion classification.

7 Work Division

The work on this project was divided between the team members, Danny and Illya, as follows:

- Both Danny and Illya were responsible for data collection and analysis, data preprocessing, and feature engineering.
- Illya focused on implementing the NLP model.
- Danny and Illya worked on the NLP model enhanced with GloVe word embeddings.

8 Learning Experience

This project provided valuable insights into the practical application of natural language processing (NLP) techniques and machine learning models. Through the process of data collection, preprocessing, and feature engineering, we gained hands-on experience in handling real-world datasets and preparing them for model training. Additionally, we deepened our understanding of the importance of text preprocessing, including tokenization, stopword removal, and embedding techniques, which play a crucial role in improving model performance.

Working with different variations of NLP models, such as a basic model and one with GloVe word embeddings, also allowed us to explore the impact of advanced techniques like word embeddings in improving the accuracy of text classification tasks. We learned how to fine-tune models and assess their performance using various evaluation metrics, helping us gain a clearer understanding of model effectiveness. Overall, this project strengthened our problem-solving and collaboration skills, as we navigated challenges together and leveraged each other's expertise to successfully complete the project.