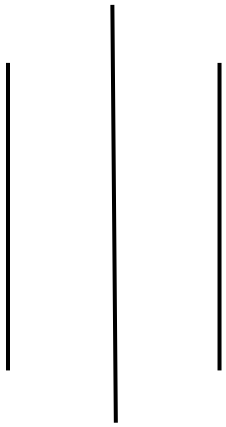
TRIBHUVAN UNIVERSITY

Institute of Science and Technology

Central Department of Computer Science and Information Technology
Kirtipur, Kathmandu, Nepal



Seminar Report on:

"Leveraging RNNs and LSTMs for Emoji Prediction"

In Partial fulfillment of the requirement for Master's Degree in Computer Science and Information Technology(M.Sc.CSIT), 1st Semester

Under the Supervision of

Dr. Dhiraj Kedar Pandey

Submitted To:

Central Department of Computer Science and Information Technology

Submitted By:

Priya Shrestha

(Roll No:24/081)

November 2025

TRIBHUVAN UNIVERSITY

Institute of Science and Technology

SUPERVISOR'S RECOMMENDATION

This is to certify that Ms. Priya Shrestha (Roll no. 24/081) has successfully completed a seminar report on **"Leveraging RNNs and LSTMs for Emoji Prediction"** under my supervision. This seminar is a partial fulfilment of the requirements for the degree of Master of Science in Computer Science and Information Technology(M.Sc.CSIT). The seminar paper demonstrates a comprehensive understanding of the subject matter and contributes to the academic discourse in the field of neural networks. I hereby, declare that this seminar report be processed for evaluation and consideration as part of the first semester curriculum.

_____

Supervisor

Dr. Dhiraj Kedar Pandey

Central Department of Computer Science and Information Technology

Kirtipur, Kathmandu, Nepal

TRIBHUVAN UNIVERSITY

Institute of Science and Technology

LETTER OF APPROVAL

This is to certify that this seminar report prepared by Priya Shrestha [8115034] entitled "**Leveraging RNNs and LSTMs for Emoji Prediction**" in partial fulfillment for the requirement of Master's Degree of Computer Science and Information Technology has been well studied. In our opinion, it is satisfactory in scope and quality as a seminar for the required degree.

Evaluation Committee

…… **Asst. Prof. Sarbin Sayami** Head of Department Central Department of CSIT	…… **Dr. Prof. Dhiraj Kedar Panday** Supervisor Central Department of CSIT
…… **External Examiner**	…… **Internal Examiner**

ACKNOWLEDGEMENT

The success and final outcome of this report required a lot of guidance and assistance from many people and I am very fortunate to have got this all along the completion. I am very glad to express my deepest sense of gratitude and sincere thanks to my highly respected and esteemed supervisor **Dr. Dhiraj Kedar Pandey,** Central Department of Computer Science and Information Technology for his valuable supervision, guidance, encouragement, and support for completing this report.

I am also thankful to **Asst. Prof. Sarbin Sayami (HOD of Central Department of Computer Science and Information Technology)** for his constant support throughout the period. His unwavering commitment to academic excellence has fostered an enriching learning environment for all students within the department.

Furthermore, with immense pleasure, I submit by deepest gratitude to the Central Department of Computer Science and Information Technology, Tribhuvan University, and all the faculty members of CDCSIT for providing the platform to explore the knowledge of interest.

Last but not the least, I would like to thank everyone who took participated in this report whether directly or indirectly. Your contributions have been instrumental in its success. Finally, I want to express our gratitude towards all the teachers, teammates and companions who greatly contributed and fueled my determination for the successful completion of this report.

Priya Shrestha [24/081]

ABSTRACT

Emojis play a vital role in modern digital communication, offering a visual and emotional layer to text-based conversations. With the growing popularity of social media platforms and messaging apps, accurately predicting emojis based on text has become a valuable application in natural language processing. The burgeoning field of emoji prediction using Natural Language Processing (NLP) and deep learning techniques is explored, particularly focused on Recurrent Neural Network (RNNs) and Long Short-Term Memory (LSTM) Networks. This seminar report uses Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) networks, a type of recurrent neural network capable of learning long-term dependencies in sequential data. The objective is to enhance digital communication by accurately predicting emojis based on text inputs, thus improving user engagement and expressiveness.

The LSTM model effectively addresses the complexities of emoji prediction including the nuances of informal language and the challenge of interpreting emotions. However, challenges such as dataset imbalance and the inherent complexity of capturing diverse emotional expressions remain. The model is implemented using Python and libraries like pandas, TensorFlow, and Keras. The model's performance is evaluated using metrics such as accuracy, precision, recall, and F1-score. Recommendations for future enhancements and research include addressing dataset imbalances and incorporating advanced techniques like attention mechanisms.

Keywords: *Emojis, Natural Language Processing (NLP), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Tokenization, Emoji Prediction*

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

The significance of numerous abbreviations and acronyms used throughout the report is described in the following lists.

Abbreviations	**Acronyms**
CSV	COMMA – SEPARATED VALUE
GLOVE	GLOBAL VECTOR REPRESENTATION
LSTM	LONG SHORT-TERM MEMORY
NLP	NATURAL LANGUAGE PROCESSING
NN	NEURAL NETWORK
RNN	RECURRENT NEURAL NETWORK

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Introduction

In today's digital era, expressive communication has become the dominant form of online interaction. Social media usage has grown to unprecedented levels, with more than 5 billion individuals actively engaging on platforms in 2024 [1]. Yet, plain text often falls short in capturing the full range of human emotions. Emojis—small graphical icons—have emerged as a powerful medium of digital expression, acting as visual symbols to convey emotions, sentiments, and context beyond plain words [2].

With the rapid expansion of platforms such as Twitter, Facebook, and Instagram, emojis have become an integral part of online communication, bridging cultural and linguistic gaps [3]. The concept of emojis originated in 1999 when Japanese mobile carrier NTT DoCoMo introduced the first set of 176 emojis. The term "emoji" itself comes from the Japanese words "*e"* (picture) and "*moji"* (letter). Since then, emojis have evolved into a global digital language, gaining widespread adoption on iOS in 2011 and WhatsApp in 2013 [4].

At the heart of this project lies the **Long Short-Term Memory (LSTM) network**, a type of recurrent neural network specifically designed to capture long-term dependencies in sequential data. By leveraging pre-trained embeddings such as **GloVe**, the model can represent text in a semantic vector space, enabling it to capture context beyond individual keywords [5]. Earlier approaches—such as rule-based systems and probabilistic models—struggled with ambiguity and contextual understanding, while deep learning methods, particularly LSTMs, have demonstrated superior performance by modeling sequential dependencies and richer semantics [6].

This project focuses on designing, implementing, and evaluating an **LSTM-based emoji prediction system** trained on sentences labeled with corresponding emojis. By experimenting with word embeddings and sequential modeling, the study investigates how effectively LSTMs capture sentence-level semantics for emoji prediction. The findings not only highlight the potential of LSTMs in expressive communication tasks but also contribute to broader research in **sentiment analysis, multimodal communication, and contextual language understanding** [7].

1.2 Problem Statement

Prediction of emojis, despite its apparent simplicity, presents significant challenges due to the complexity and ambiguity of natural language. Firstly, determining the most contextually appropriate emoji from subtle and nuanced text requires understanding semantic meaning, emotional tone, and informal expressions, making it a non-trivial classification problem. Secondly, keyword-matching approaches based on the language used in chats often fail to capture context, leading to inaccurate or irrelevant predictions due to the prevalence of colloquial expressions and informal language.

Additionally, the imbalance in emoji usage across datasets where certain emojis appear far more frequently than others—further complicates model training, potentially biasing predictions toward popular emojis. Short or ambiguous sentences, sarcasm, and cultural variations in emoji interpretation add to the challenge. Therefore, there is a critical need for models, such as LSTM-based deep learning networks, that can effectively learn sequential dependencies, capture semantic and emotional nuances, and make accurate emoji predictions while maintaining computational efficiency. The primary challenge in emoji prediction is the lack of context-specific user data. Since the same emoji can convey different meanings depending on the user, having user-specific information makes it easier to generate accurate predictions.

1.3 Objectives

The main objective of this seminar includes:

- To accurately predict the most suitable emoji for a given text by capturing semantic meaning and emotional context using both RNN and LSTM model.
- To explore and evaluate the effectiveness and performance of LSTM networks in learning sequential patterns and handling informal language, slang, and contextual ambiguity for reliable emoji prediction using metrices.

1.4. Report Organization

The remainder of this report is organized into five chapters:

- **Chapter 1: Introduction** Provides an overview of the research, including the background, problem statement, objectives and the structure of the report.

- **Chapter 2: Literature Review** Surveys prior research on emoji prediction, sentiment analysis, and deep learning approaches such as RNNs and LSTMs. It also highlights existing research gaps and challenges.

- **Chapter 3: Methodology** Describes the dataset, preprocessing techniques, model architecture, word embeddings, training procedure, and evaluation metrics used in the project.

- **Chapter 4: Implementation and Results** Details the implementation of the LSTM-based emoji prediction model, system architecture, tools used, and presents experimental results with analysis.

- **Chapter 5: Conclusion and Future Work** Summarizes key findings, discusses model performance, limitations, and provides recommendations for future improvements in emoji prediction systems.

In addition, the report includes:

- **References** A complete list of cited works.
- **Appendix** Contains supporting code, datasets, and supplementary material.

CHAPTER 2: LITERATURE REVIEW

2.1 Background Study

Emojis have become an essential part of digital communication, adding a visual and emotional layer to text messages that plain words often cannot convey. They help express moods, feelings, and subtle nuances, making online conversations more engaging and expressive [1][2]. Predicting the right emoji based on text is a challenging task in Natural Language Processing (NLP) because it requires understanding context, emotion, and meaning in often short, informal, or ambiguous sentences.

Early approaches to emoji prediction relied on simple keyword matching or rule-based methods. These methods were limited because they often failed to capture context, slang, abbreviations, or informal expressions commonly used in chats and social media [3]. Probabilistic models improved on this by estimating the likelihood of emojis for certain words or phrases, but they still struggled with sequential dependencies and emotional subtleties.

Deep learning, especially Long Short-Term Memory (LSTM) networks, offers a better solution. LSTMs can learn patterns in sequences of words, understand context, and capture the meaning of sentences, which is crucial for accurate emoji prediction [4][5]. Using pre-trained word embeddings like GloVe further improves the model's understanding by mapping words into semantic vector spaces, allowing it to recognize subtle differences in meaning.

Despite these advancements, challenges remain, including dataset imbalance, sarcasm, cultural differences in emoji usage, and short or ambiguous text [6][7]. Still, LSTM-based models provide a powerful framework for predicting emojis, helping make digital communication more expressive, accurate, and engaging.

Moreover, studies demonstrate that LSTM-based models perform well across different languages and writing styles, such as in emoji prediction for Arabic microblogs using deep recurrent neural networks [12]. Recent research explores improvements such as emoji embeddings, sentiment-aware emoji prediction, and attention mechanisms that help the model focus on emotionally relevant words [14]. Context-aware models show that emoji usage patterns vary across users and contexts, indicating that personalization further improves prediction results [15].

In summary, emoji prediction is a complex NLP task that combines sequential language modeling, semantic understanding, and emotion recognition. LSTM-based deep learning models, enhanced with pre-trained embeddings, provide a robust framework to tackle these challenges, enabling more accurate and context-aware emoji predictions, which ultimately enhance expressiveness and engagement in digital communication.

2.2. Literature Review

Extensive research has been conducted on emoji prediction, demonstrating a variety of approaches and their relative effectiveness. Early methods primarily relied on **rule-based or keyword-matching techniques**, where specific words or phrases in a sentence were mapped to corresponding emojis [1]. While computationally simple, these approaches often failed to capture context, colloquial expressions, or the nuanced emotional meaning of text, leading to inaccurate predictions [2].

Machine learning methods, particularly **probabilistic models** such as Naive Bayes or logistic regression, were introduced to improve prediction by estimating the likelihood of an emoji given the text [3]. These models incorporated text features like bag-of-words or TF-IDF vectors, offering better performance than rule-based methods. However, they were limited in handling **sequential dependencies**, contextual meaning, and long-range semantic relationships in sentences [4].

The advent of **deep learning**, specifically **Recurrent Neural Networks (RNNs)** and **Long Short-Term Memory (LSTM) networks**, marked a significant improvement in emoji prediction. LSTMs can model sequential patterns and capture long-term dependencies in text, allowing for a deeper understanding of context, emotion, and semantic nuances [5]. Integrating **pre-trained word embeddings** such as GloVe further enhanced these models by providing semantic representations of words, which enabled more accurate emoji predictions even in informal or ambiguous text [6].

Recent advancements include **attention mechanisms and multimodal approaches**, where models consider additional context such as images, social interactions, or emoji co-occurrence patterns [7]. Attention-based LSTM models allow the network to focus on the most relevant words in a sentence for predicting emojis, improving accuracy and handling complex sentence structures [8] Despite these advances, challenges such as **dataset imbalance, sarcasm, short text, cultural differences, and rare emoji predictions** remain prevalent, requiring careful model design and evaluation.[9]

Collectively, these research efforts highlight emoji prediction as a rich testbed for evaluating NLP techniques that must balance semantic understanding, contextual reasoning, and efficient sequence modeling [10]. Further exploration of deep learning architectures, attention mechanisms, and multimodal strategies is expected to enhance the accuracy and applicability of emoji prediction models in real-world digital communication [11]

In addition, transformer-based architectures like BERT and GPT have shown strong capabilities in contextual text understanding and are increasingly being applied to emoji prediction tasks [12]. Hybrid approaches combining sentiment analysis with emoji prediction have demonstrated improved accuracy by aligning emotional tone with appropriate emoji output [13]. Researchers have also explored data augmentation and transfer learning techniques to address dataset imbalance and improve generalization in low-resource scenarios [14]. Finally, recent studies highlight the need to incorporate personalization and user preference modeling, as emoji usage patterns vary between individuals and cultures, making personalized prediction a promising future direction [15].

CHAPTER 3: METHODOLOGY

3.1 Conceptual Framework

The conceptual framework for this research is structured to systematically develop, train, and evaluate deep learning models for emoji prediction. The process follows several sequential steps as outlined below:

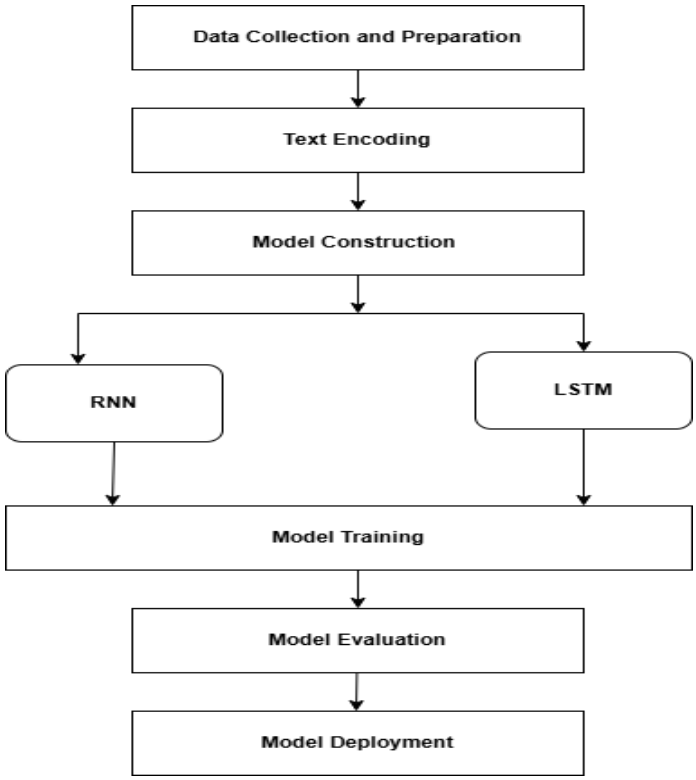


Figure 3.1: Flowchart of proposed model

3.2 Data Collection and Preparation

	0	1
0	never talk to me again	3
1	I am proud of your achievements	4
2	It is the worst day in my life	3
3	Miss you so much	0
4	hello sunshine	4
...	...	...
129	What is wrong with you	3
130	I love you	0
131	great job	4
132	hi where are you	1
133	sayonara	1

Figure 3.2: Data Collection and Preparation

Begins by gathering textual dataset collected from various sources such as publicly available datasets, web scraping, or user input data along with associated emojis to serve as the training and testing datasets. The collected data may contain noise, missing values, or irrelevant content. Therefore, preprocessing steps such as cleaning unwanted characters, tokenization, and normalization are performed. The main objective of this step is to ensure that the data is consistent and ready for encoding and further processing.

3.2.1 Text Encoding

Machine learning models cannot process raw text, so encoding is required. The cleaned text is then tokenized breaking it into individual words or subwords. These tokens are transformed into sequences of numbers and padded to maintain equal input length. Pre-trained word embeddings (e.g., GloVe) are then used to convert words into dense vector representations that capture contextual meaning. These embeddings serve as input to the neural network.

3.2.2 Model Construction

In this step, different types of recurrent neural network architectures are designed and experimented with:

- **RNN:** Basic recurrent architecture that processes sequences but struggles with long-term dependencies.
- **LSTM (Long Short-Term Memory):** An improved advanced recurrent neural network that introduces memory gates to handle long-range dependencies and overcome the vanishing gradient problem.

These models are built with layers such as Embedding, LSTM, Dense, and Dropout for regularization.

3.2.3 Model Training

The prepared dataset is split into training and validation sets. During training, the model learns the relationship between text sequences (input) and emojis (output). Optimization techniques such as Adam and loss functions like categorical cross-entropy are used to minimize error. Hyperparameters such as batch size, number of epochs, and learning rate are tuned for better performance. The objective of this step is to minimize the error between predicted and actual output while improving model accuracy.

3.2.4 Model Evaluation

Once training is completed, each model is evaluated using metrics like accuracy, precision, recall, and F1-score. Confusion matrices are also analyzed to examine prediction performance across emoji categories. This step helps to determine which architecture performs best for emoji sentiment classification.

3.2.5 Model Deployment

The best-performing model is saved and deployed so it can be used for real-time emoji prediction. During deployment, the model takes a text input from the user, processes it through the same encoding steps, and outputs the most suitable emoji based on the learned patterns. This enables integration into applications such as chat systems or sentiment-aware messaging platforms.

3.3 Algorithm Description

This section describes the four deep learning models implemented for the task of emoji prediction from textual input. The aim is to predict an emoji that best represents the meaning, mood, or sentiment expressed in a given sentence. Each model processes the input text as a sequence of words, learns contextual patterns, and outputs the most probable emoji class. Let the input text be represented as $X=[x_1, x_2, x_3,\ldots,x_T]$

where x_t is the word embedding at time step t, and T is the sequence length. Each model computes a sequence of hidden states h_t, which encode contextual information, and then passes the final state to a softmax classifier for emoji prediction.

a. RNN (Recurrent Neural Network)

- **Concept:**

Recurrent Neural Network are a type of neural network where the output from previous step is fed as input to the current step. RNN maintains a hidden state ("memory") that stores information from previous inputs, allowing the network to use past context when making predictions. It processes text word by word, keeping track of information through a hidden state that gets updated at each step. This hidden state represents what the network remembers about the sentence so far. Each new word updates the hidden state based on

both the current word and the previous state. After processing all words, the final state is used to predict which emoji best represents the entire sentence.

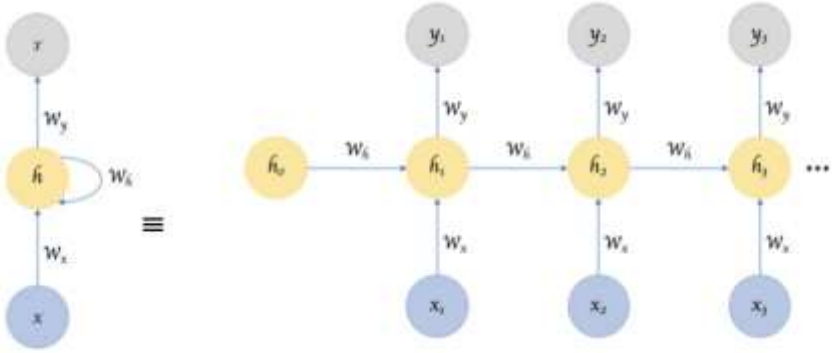


Figure 3.3.1: RNN Model

- **Mathematical Operation:**

At each time step t:

$$\mathbf{h_t = \tanh(W_x X_t + W_h h_{t-1} + b_h)}$$

$$\hat{y} = \mathbf{softmax}(W_y h_t + b_y)$$

where,

- W_x: Weight matrix applied to the current input x_t (input to hidden)
- W_h: Weight matrix applied to the previous hidden state h_{t-1}
- X_t : Input at time step t (current word embedding vector)
- W_y : Weight matrix applied to the final hidden state for prediction
- b_h : bias added during hidden state calculation
- b_y: bias added before output layer,
- h_t is the hidden state, at time step t,
- h_{t-1}: Previous hidden state
- $\hat{y}$ is the predicted emoji probability distribution.

- **Model Flow:**

Input text → Embedding → RNN Layer → Dense Output Layer → Softmax → Predicted Emoji

b. LSTM (Long Short-Term Memory Network)

- **Concept:**

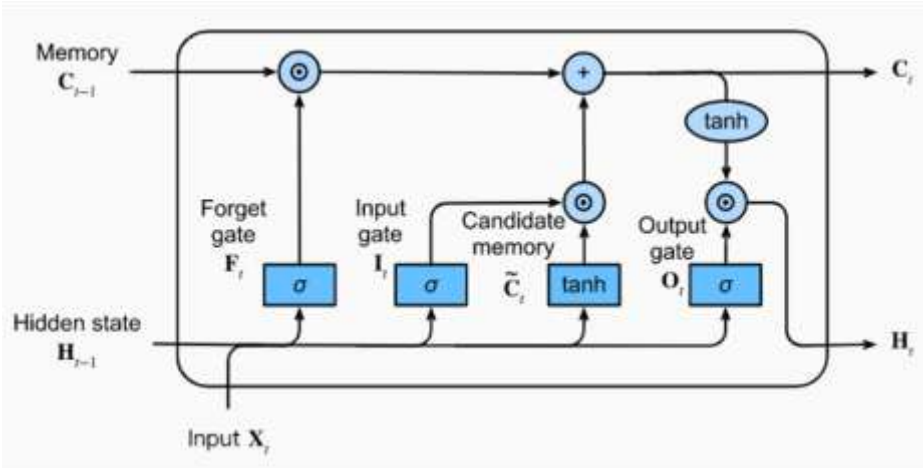


Figure 3.3.2: LSTM Model

LSTM (Long Short-Term Memory) is an improved variant of Recurrent Neural Networks (RNN) designed to learn and retain long-term dependencies in sequential data. Traditional RNNs struggle with the *vanishing gradient problem*, which makes it difficult for them to remember information over long sequences. LSTMs overcome this limitation by introducing a **memory cell** that maintains two types of states:

- **Cell state** (long-term memory) stores important information across many time steps.
- **Hidden state** (short-term memory) holds information relevant to the current step.

Gates:

LSTMs use three internal "gates":

- **Forget Gate:** Determines what information must be remembered and what can be forgotten from previous step.
- **Input Gate:** Determines what new information to store.
- **Output Gate:** Determines the next hidden state value.

- **Mathematical Operation:**

For each time step t:

Forget gate : $F_t=\sigma(X_t W_f + H_{t-1} U_f)$

Input gate : $I_t = \sigma (X_t W_i + H_{t-1} U_i)$

$H'_t = \tanh(X_t W_c + H_{t-1} U_c)$ [New candidate information]

$C_t = \sigma (C_{t-1} F_t + I_t H'_t)$ [New cell state]

Output gate : $O_t = \sigma (X_t W_o + H_{t-1} U_o)$

$H_t = \tanh(C_t * O_t)$ [New information]

Finally, the output emoji is predicted as: $\hat{y} = \text{softmax}(W_y H_t + b_y)$

where

- $W_f, U_f, W_c, U_c, W_i, U_i, W_o, U_o$: Weight matrices
- X_t : Input at time step t (current word embedding vector)
- b_y: bias added before output layer,
- H_t is the hidden state, at time step t,
- h_{t-1}: Previous hidden state
- C_t : cell state
- H'_t: new candidate information
- $\hat{y}$ is the predicted emoji probability distribution.

- **Model Flow:**

Input → Embedding → LSTM Layer → Dense Output Layer → Softmax → Predicted Emoji

Chapter 4: Implementation and Result Analysis

4.1 Implementation

4.1.1 Implementation Tools Used

In the implementation of this project, the following tools and libraries were utilized. Python (the primary programming language), Keras (a high-level neural networks API built on TensorFlow), NumPy (for numerical computations), Pandas (for data manipulation and analysis) Matplotlib (for plotting and visualization)

4.1.2 Implementation Details

Implementation details include:

1. Import Libraries and Read Data

In the implementation plan, the necessary libraries like pandas, numpy, tensorflow , keras , embedding, emojis, RNN, LSTM, etc were imported. Then '.csv' file holding texts and their corresponding integer values for emojis were read in figure 2.

2. Install and Import Emoji Library

Install the emoji library and import it to handle emoji data in figure 3.

3. Define Emoji Dictionary and Label Mapping Function

Define a dictionary to map labels to the emojis and create a function to convert labels to emoji characters in figure 4.

4. Prepare Data

Extract input texts (X) and labels(Y) from the DataFrame in figure 5.

5. Tokenization and Pad Sequences

The input text data (x) are converted into tokens using a Keras tokenizer (texts to sequences function) and y data into one hot encoding using categorical function. The length of the sentences is fixed using maxlen function and padding truncate the values in figure 6.

6. Perform Embedding

Assuming pre-trained embeddings, prepare the embedding matrix to turns words (or tokens) into dense vector in figure 7.

7. Build and train the model

Construct the both model RNN and LSTM in figure 8.

8. Predict on new data

Prepare test data, tokenize, padsequences and make predictions using the trained model in figure 9.

4.2 Result Analysis.

Evaluation of the emoji prediction using LSTM model is based on the confusion matrix.

a. RNN

```
Text: I got approved
Predicted: ⭐
Actual: ⭐
------
Text: I love reading books
Predicted: 😃
Actual: ❤️
------
Text: hi where are you
Predicted: ❤️
Actual: 👋
------
Text: good job
Predicted: 😃
Actual: 😃
------
Text: so bad you cannot keep words
Predicted: 😩
Actual: 😩
------
Correct predictions: 3
Incorrect predictions: 2
```

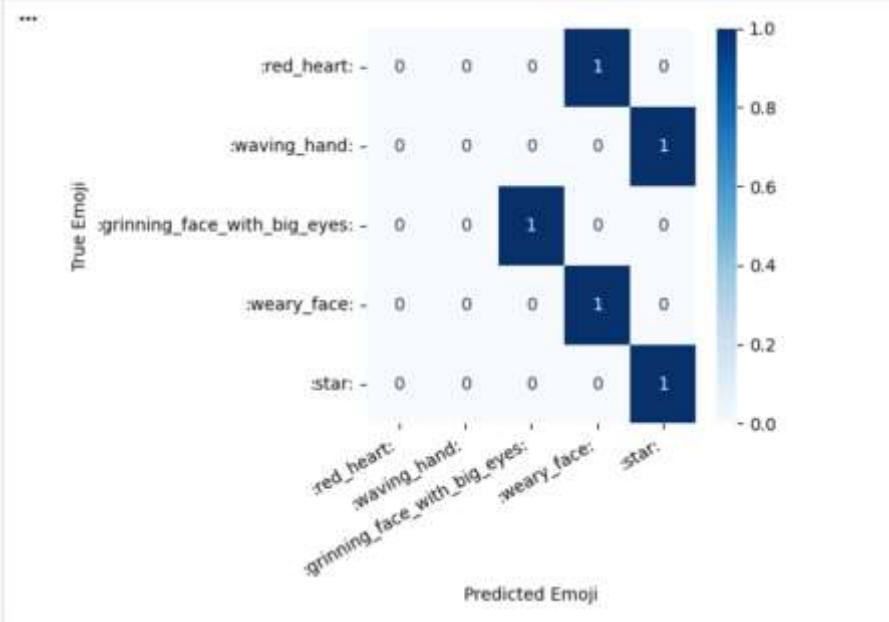


Figure 4.2.1: Emoji Prediction Confusion Matrix(RNN)

b. LSTM

```
Text: I got approved
Predicted: ⭐
Actual: ⭐
------
Text: I love reading books
Predicted: 😀
Actual: ❤️
------
Text: hi where are you
Predicted: 👋
Actual: 👋
------
Text: good job
Predicted: 😀
Actual: 😀
------
Text: so bad you cannot keep words
Predicted: 😩
Actual: 😩
------
Correct predictions: 4
Incorrect predictions: 1
```

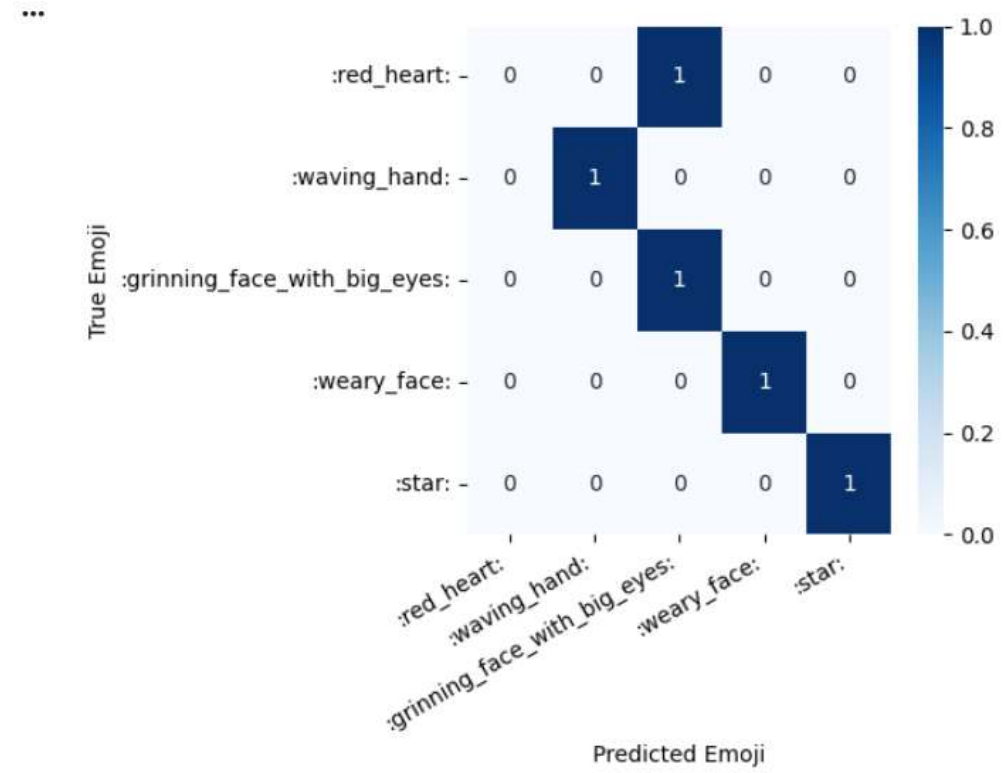


Figure 4.2.2: Emoji Prediction Confusion Matrix (LSTM)

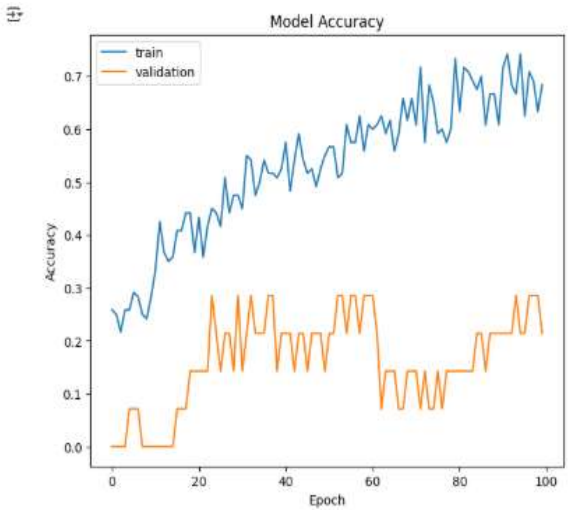
Performance Metrics and Model Performance

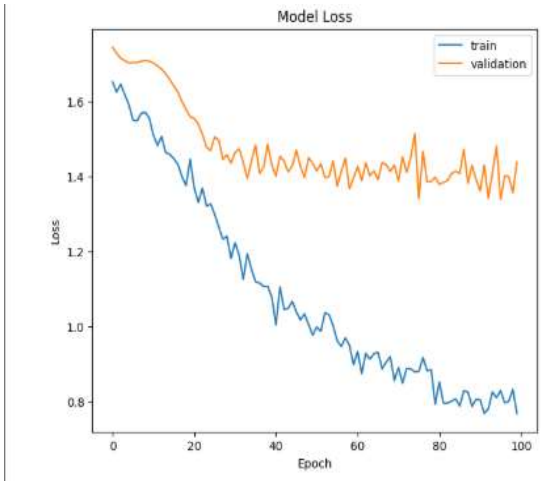
a. RNN

```
Test Accuracy: 0.6000
Precision (macro): 0.5000
Recall (macro): 0.6000
F1-score (macro): 0.5333
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00         1
           2       0.50      1.00      0.67         1
           3       1.00      1.00      1.00         1
           4       1.00      1.00      1.00         1

    accuracy                           0.60         5
   macro avg       0.50      0.60      0.53         5
weighted avg       0.50      0.60      0.53         5
```





b. LSTM

```
Test Accuracy: 0.8000
Precision (macro): 0.7000
Recall (macro): 0.8000
F1-score (macro): 0.7333
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       0.00      0.00      0.00         1
           2       1.00      1.00      1.00         1
           3       1.00      1.00      1.00         1
           4       0.50      1.00      0.67         1

    accuracy                           0.80         5
   macro avg       0.70      0.80      0.73         5
weighted avg       0.70      0.80      0.73         5
```

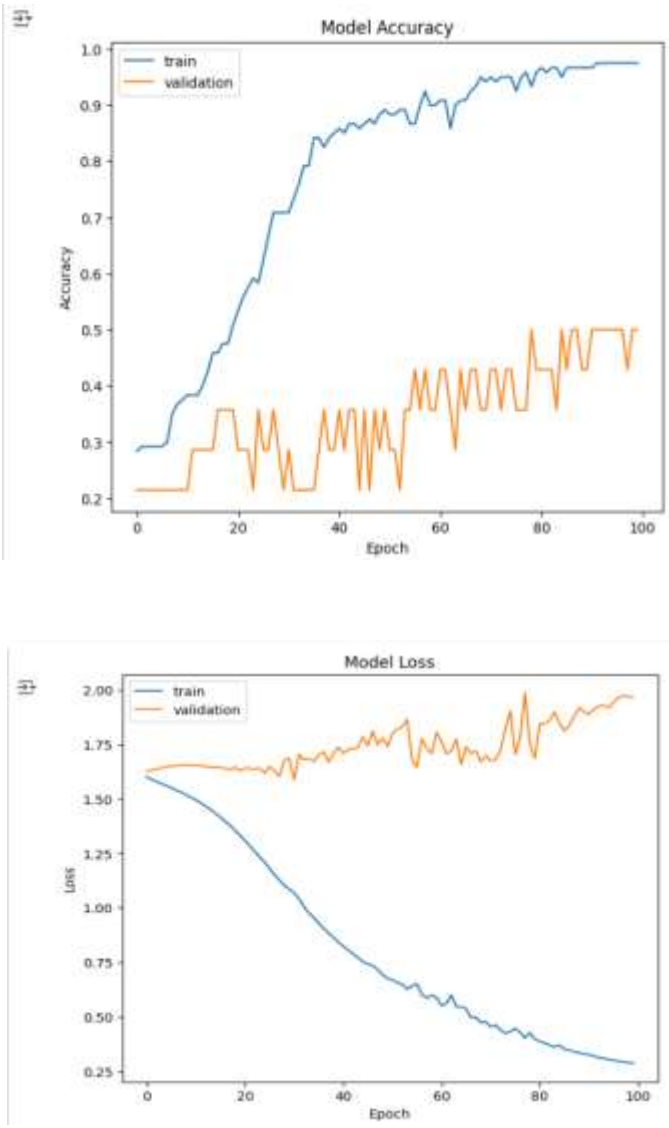


Figure 4.2.3: Performance Metrics and Performance Model

CHAPTER 5: CONCLUSION AND FUTURE RECOMMENDATIONS

5.1. Conclusion

In conclusion, two deep learning architectures **RNN** and **LSTM (Long Short-Term Memory)** were implemented and compared for emoji prediction from text sequences. Both models were trained using labeled text emoji pairs and evaluated based on training accuracy, validation accuracy, loss, and classification performance. The results show that while the **RNN** can learn basic sequential patterns, it struggles to maintain long-term dependencies in the text. This leads to lower accuracy and unstable validation performance, especially as sentences become longer.

In contrast, the **LSTM model achieved significantly better performance**, reaching higher training and test accuracy, and showing improved recall and F1-score across multiple emoji classes. LSTM's internal memory mechanism (input, forget, and output gates) helps it effectively preserve contextual information across sequences, making it more suitable for understanding sentiment and tone required for emoji prediction.

However, overfitting was observed in LSTM training as the training accuracy increased while validation accuracy fluctuated. This indicates that while LSTM learns the training data well, additional regularization or more diverse data is needed to improve generalization.
Moreover, the LSTM model performed significantly better, achieving higher accuracy and improved classification results across multiple emoji classes. LSTM's gating mechanism allowed it to remember important information over longer sequences, making it more effective for understanding emotional tone and sentiment in text. Overall, LSTM proved to be the superior approach for emoji prediction, while RNN was faster but less accurate and less reliable.

5.2. Future Recommendations

Based on the observations from this study, the following recommendations can guide further research and implementation:

1. **Use larger and balanced datasets**

   The model's performance may be affected by the imbalance in the dataset, where certain emojis are underrepresented.

2. **Try advanced models**

   Experiment with **Bi-LSTM, GRU, or Transformer models (BERT, GPT)** for improved contextual understanding.

3. **Apply regularization techniques**

   Regularization Techniques such as dropout and early stopping must be applied to prevent overfitting in LSTM.

4. **Optimize hyperparameters**

   Hyperparameters such as learning rate, batch size, number of neurons must be optimized to improve overall performance and training efficiency.

5. **Incorporate user context and sentiment awareness**

   Analyze the emotion of the text, not just the words. By detecting whether the message is positive, negative, or neutral, the model can choose a more accurate emoji.

REFERENCES

[1] R. Ranjan, P. Yadav, "Emoji prediction using LSTM and Naive Bayes," in TENCON 2021-2021 IEEE Region 10 Conference (TENCON), 2021, pp. 284–288.

[2] Kone, V., et al, "Emoji prediction using bi-directional LSTM," in ITM Web of Conferences, 2023, pp. 02004.

[3] Zhang, L., et al, "Emoji prediction: A transfer learning approach," in *Future of Information and Communication Conference*, 2020, pp. 864–872.

[4] Shaik, A., et al. "Recurrent neural network with emperor penguin-based Salp swarm (RNN-EPS2) algorithm for emoji based sentiment analysis," in *Multimedia Tools and Applications*, vol. 83, no. 12, pp. 35097–35116, 2024.

[5] H. Raj, S. Balachandran. "Future emoji entry prediction using neural networks," in *Journal of Computer Science*, vol. 16, no. 2, pp. 150–157, 2020.

[6] Zhou, C., et al. "A C-LSTM neural network for text classification," in *arXiv preprint arXiv:1511.08630*, 2015.

[7] A. Sherstinsky. "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," in *Physica D: Nonlinear Phenomena*, vol. 404, pp. 132306, 2020.

[8] Yu, Y., et al. "A review of recurrent neural networks: LSTM cells and network architectures," in *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[9] Bai, Q., et al. "A systematic review of emoji: Current research and future perspectives," in *Frontiers in psychology*, vol. 10, pp. 476737, 2019.

[10] Liu, C., Fang, F., Lin, X., Cai, T., Tan, X., Liu, J., & Lu, X. (2021). Improving sentiment analysis accuracy with emoji embedding. *Journal of Safety Science and Resilience, 2(4), 246–252.*

[11] Mittal, M., Arora, M., & Pandey, T. (2019). Emoticon prediction on textual data using stacked LSTM model. In *International Conference on Communication and Intelligent Systems* (pp. 259–269).

[12] Al-Azani, S., & El-Alfy, E.S. (2018). Emojis-based sentiment classification of Arabic microblogs using deep recurrent neural networks. In *2018 international conference on computing sciences and engineering (ICCSE)* (pp. 1–6).

[13] Hussain, S., Alhussayni, K., & Obayes, H. (2024). Iraqi Dialect Emoji Prediction Based on Deep CNN-LSTM Architecture. In *National Conference on New Trends in Information and Communications Technology Applications* (pp. 128–141).

[14] Lou, Y., Zhang, Y., Li, F., Qian, T., & Ji, D. (2020). Emoji-based sentiment analysis using attention networks. *ACM Transactions on asian and low-resource language information processing (TALLIP), 19(5), 1–13.*

[15] Gupta, A., Bhatia, B., Chugh, D., Himabindu, G., Sethia, D., Agarwal, E., Sani, D., & Garg, S. (2021). Context-aware emoji prediction using deep learning. In *International Conference on Artificial Intelligence and Speech Technology* (pp. 244–254).

APPENDIX

```
import numpy as np
import pandas as pd
import tensorflow as tf
import random

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, SimpleRNN, Embedding, Bidirectional, Dropout

from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.utils import to_categorical


import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, recall_score, f1_score, precision_score

#Loading the dataset
data=pd.read_csv('/content/train_emoji.csv', header=None)
data = data.drop(data.columns[[2, 3]], axis=1)
data.head(6)
```

Figure 2: Import Libraries and Read Data

```
!pip install emoji
import emoji
```

Figure 3: Install and Import Emoji Library

```
# Creating dictionary for some emoji's, consisting of key and value
emoji_dict = {
    0: ':red_heart:',
    1: ':waving_hand:',
    2: ':grinning_face_with_big_eyes:',
    3: ':weary_face:',
    4: ':star:'
}
emoji_dict
```

```
def label_to_emoji(label):
    return emoji.emojize(emoji_dict.get(label), language='alias')

for i in emoji_dict.keys():
    print(label_to_emoji(i))
```

❤️
👋
😃
😩
⭐

Figure 4: Define Emoji Dictionary and Label Mapping Function

```
#Prepare data(text pre processing)
x = data[0].values
y = data[1].values
x, y
# len(x), len(y)
```

Figure 5: Prepare Data

```
#Initialize tokenizer
tokenizer = Tokenizer()
tokenizer
```

```
<keras.src.legacy.preprocessing.text.Tokenizer at 0x780894319820>
```

```
tokenizer.fit_on_texts(x_cleaned)
word_to_index =tokenizer.word_index
# print(tokenizer.word_index)
len(word_to_index)
```

```
262
```

```
x_tokens =tokenizer.texts_to_sequences(x_cleaned)
x_tokens
```

```
maxlen=0
for data in x_tokens:
  maxlen = max(maxlen, len(data))
maxlen
```

```
10
```

```
#Padded sequences
x_train = pad_sequences(x_tokens, maxlen=maxlen, padding='post', truncating='post')
x_train
```

```
array([[ 86,  87,   9, ...,   0,   0,   0],
       [  1,   6,  52, ...,   0,   0,   0],
       [ 17,   3,   4, ...,  55,   0,   0],
       ...,
       [ 50,  29,   0, ...,   0,   0,   0],
       [261,  85,   8, ...,   0,   0,   0],
       [262,   0,   0, ...,   0,   0,   0]], dtype=int32)
```

```
y_train = to_categorical(y)
num_classes = len(np.unique(y_train))
print(num_classes)
y_train
```

```
print(x_train.shape)
print(y_train.shape)
```

```
(134, 10)
(134, 5)
```

Figure 6: Tokenization and Pad Sequences

```
#Get Embeddings :An embedding is a dense vector (usually 50-300 dimensions) representing a word.
#Each word is mapped to a vector of real numbers.
file = open('/content/glove.6B.100d.txt', 'r', encoding='utf8')
content= file.readlines()
file.close()
```

```
#creating embedding dictionary from glove file
embedding = {}

for line in content:
    line = line.split()
    embedding[line[0]] =np.array(line[1:], dtype =float)

print("Total words in embedding:", len(embedding))
print("First 10 words:", list(embedding.keys())[:10])
print("\n")

# Printing first 5 words with first 10 vectors values
for i, (word, vector) in enumerate(embedding.items()):
    print(word, vector)
    if i == 4:
        break
```

```
Total words in embedding: 400000
First 10 words: ['the', ',', '.', 'of', 'to', 'and', 'in', 'a', '"', "'s"]


the [-0.038194 -0.24487   0.72812  -0.39961   0.083172  0.043953 -0.39141
  0.3344   -0.57545   0.087459  0.28787  -0.06731   0.30906  -0.26384
 -0.13231  -0.20757   0.33395  -0.33848  -0.31743  -0.48336   0.1464
```

```
embed_size =100
embedding_matrix =np.zeros((len(word_to_index)+1,embed_size))
embedding_matrix.shape
```

```
(263, 100)
```

```
#creating embedding matrix
for word, i in word_to_index.items():
    embed_vector = embedding.get(word)
    if embed_vector is not None:
        embedding_matrix[i] = embed_vector
    else:
        embedding_matrix[i] = np.random.normal(scale =0.6, size=(embed_size,))
```

```
embedding_matrix
```

```
array([[ 0.      ,  0.      ,  0.      , ...,  0.      ,  0.      ,
         0.      ],
       [-0.046539,  0.61966 ,  0.56647 , ..., -0.37616 , -0.032502,
         0.8062  ],
       [-0.49886 ,  0.76602 ,  0.89751 , ..., -0.41179 ,  0.40539 ,
         0.78504 ],
       ...,
```

Figure 7: Perform Embedding

```
#Build the model(RNN)
embed_size = 100

model = Sequential([
  Embedding(input_dim=len(word_to_index)+1,  #embedding layer that maps token to coresponding embeding vector
  output_dim = embed_size,
  input_length = maxlen,
  weights =[embedding_matrix],
  trainable =False
),
    SimpleRNN(8,  return_sequences=False),
    Dropout(0.4),
    Dense(5, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
# history =model.fit(x_train, y_train, epochs=100)
history = model.fit(
    x_train, y_train,
    epochs=100,
    batch_size=32,
    validation_split=0.1
)
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step - accuracy: 0.7835 - loss: 0.7493 - val_accuracy: 0.1429 - val_loss: 2.1946
Epoch 94/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.8438 - loss: 0.7269 - val_accuracy: 0.1429 - val_loss: 2.1988
Epoch 95/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 0.7615 - loss: 0.6306 - val_accuracy: 0.1429 - val_loss: 2.1978
Epoch 96/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 0.8231 - loss: 0.6659 - val_accuracy: 0.1429 - val_loss: 2.2118
Epoch 97/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.7619 - loss: 0.7338 - val_accuracy: 0.1429 - val_loss: 2.2316
Epoch 98/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.7900 - loss: 0.7082 - val_accuracy: 0.1429 - val_loss: 2.2324
Epoch 99/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.7908 - loss: 0.6981 - val_accuracy: 0.1429 - val_loss: 2.2314
Epoch 100/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.8035 - loss: 0.6889 - val_accuracy: 0.1429 - val_loss: 2.2309
```

```
#Build the model(LSTM)
embed_size = 100

model = Sequential([
  Embedding(input_dim=len(word_to_index)+1,  #embedding layer that maps token to coresponding embeding vector
  output_dim = embed_size,
  input_length = maxlen,
  weights =[embedding_matrix],
  trainable =False
),
  LSTM(units =16, return_sequences=True),
  LSTM(units= 10, return_sequences=True),
  LSTM(units = 4),
  Dense(units= 5,activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
# history =model.fit(x_train, y_train, epochs=100)
history = model.fit(
    x_train, y_train,
    epochs=100,
    batch_size=32,
    validation_split=0.1
)
```

```
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.9733 - loss: 0.3091 - val_accuracy: 0.5000 - val_loss: 1.9282
Epoch 95/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.9712 - loss: 0.3090 - val_accuracy: 0.5000 - val_loss: 1.9185
Epoch 96/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.9848 - loss: 0.2880 - val_accuracy: 0.5000 - val_loss: 1.9424
Epoch 97/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.9650 - loss: 0.3050 - val_accuracy: 0.5000 - val_loss: 1.9631
Epoch 98/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.9671 - loss: 0.3055 - val_accuracy: 0.4286 - val_loss: 1.9728
Epoch 99/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.9848 - loss: 0.2748 - val_accuracy: 0.5000 - val_loss: 1.9691
Epoch 100/100
4/4 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.9785 - loss: 0.2832 - val_accuracy: 0.5000 - val_loss: 1.9651
```

Figure 8: Build and train the model

```
import re

test = [
    "I got approved",
    "I love reading books",
    "hi where are you",
    "good job",
    "so bad you cannot keep words",
]

def clean_text(text):
    text = text.lower()                                  # lowercase
    text = re.sub(r"http\S+|www\S+", "", text)           # remove links
    text = re.sub(r"[^a-z\s]", "", text)                 # keep only letters & spaces
    text = re.sub(r"\s+", " ", text).strip()             # remove extra spaces
    return text


cleaned_test = [clean_text(t) for t in test]
test_seq =tokenizer.texts_to_sequences(cleaned_test)
X_test = pad_sequences(test_seq, maxlen=maxlen, padding="post", truncating="post")
```

```
Y_predict = model.predict(X_test)
Y_predict
```

```
Y_pred =np.argmax(Y_predict, axis = 1)
Y_pred
```

```
array([0, 0, 3, 2, 2])
```

```
for i in range(len(cleaned_test)):
  print(cleaned_test[i], label_to_emoji(Y_pred[i]))
```

```
i got approved ❤️
i love reading books ❤️
hi where are you 😍
good job 😀
so bad you cannot keep words 😀
```

```
import re

test = [
    "I got approved",
    "I love reading books",
    "hi where are you",
    "good job",
    "so bad you cannot keep words",
]

def clean_text(text):
    text = text.lower()                              # lowercase
    text = re.sub(r"http\S+|www\S+", "", text)       # remove links
    text = re.sub(r"[^a-z\s]", "", text)             # keep only letters & spaces
    text = re.sub(r"\s+", " ", text).strip()         # remove extra spaces
    return text



cleaned_test = [clean_text(t) for t in test]
test_seq =tokenizer.texts_to_sequences(cleaned_test)
X_test = pad_sequences(test_seq, maxlen=maxlen, padding="post", truncating="post")
```

```
Y_predict = model.predict(X_test)
Y_predict
```

```
Y_pred =np.argmax(Y_predict, axis = 1)
Y_pred
```

```
array([4, 0, 4, 2, 3])
```

```
for i in range(len(cleaned_test)):
  print(cleaned_test[i], label_to_emoji(Y_pred[i]))
```

```
i got approved ⭐
i love reading books ❤️
hi where are you ⭐
good job 😃
so bad you cannot keep words 😜
```

Figure 9: Predict on new data