**ASSIGNMENT HELP**

**MANUAL**



SUBMITTED

TO

VISHWAKARMA INSTITUTE OF INFORMATION TECHNOLOGY, PUNE

FOR THE SKILL AND COMPETENCY EVALUATION OF

DEEP LEARNING [ CAUA31202]

IN

**CSE AI DEPARTMENT**

BY

**Vedant Rakesh Mukhekar**

**Class: T.Y. BTech Division: A Batch: A2**

**Batch Teacher**

**Dr. ANURADHA YENKIKAR.**

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR. NO.** | **CONTENTS** | **PAGE NO.** |
| **1** | **PROBLEM STATEMENT** | **4-5** |
| **2** | **LIBRARY USED** | **5-6** |
| **3** | **THEORY** | **6-7** |
| **4** | **METHODOLOGY** | **7-14** |
| **5** | **ADVANTAGES & DISADVANTAGES** | **14-15** |
| **6** | **WORKING** | **16-17** |
| **7** | **DIAGRAM** | **17-18** |
| **8** | **CONCLUSION** | **18-19** |

### Problem Statement

The objective of this project is to implement a sentiment analysis system using **Long Short-Term Memory (LSTM)** networks or **Gated Recurrent Units (GRU)**. The goal is to classify text data—such as movie reviews, social media posts, or customer feedback—into sentiment categories (positive, negative, or neutral). By leveraging LSTM or GRU, which are well-suited for sequence data, the project aims to accurately capture the context and semantics of the text for effective sentiment classification.

### Libraries Used

* **TensorFlow/Keras**: Libraries for building and training deep learning models.
* **NumPy**: A library for numerical operations in Python.
* **Pandas**: A library for data manipulation and analysis.
* **Matplotlib**: A plotting library for visualizing training results and model performance.
* **NLTK (Natural Language Toolkit)**: A library for working with human language data, providing tools for text preprocessing.
* **scikit-learn**: A library for machine learning providing utilities for model evaluation and data processing.

### Theory

**Sentiment Analysis** is a natural language processing (NLP) task that involves determining the sentiment expressed in a piece of text. LSTMs and GRUs are types of Recurrent Neural Networks (RNNs) that excel in processing sequences of data due to their ability to maintain long-term dependencies and capture contextual information.

#### Key Concepts

* **LSTM (Long Short-Term Memory)**: A type of RNN that addresses the vanishing gradient problem, allowing it to learn long-term dependencies effectively. It includes memory cells that store information over time.
* **GRU (Gated Recurrent Unit)**: A variant of LSTM that combines the forget and input gates into a single update gate, simplifying the architecture while retaining the ability to capture dependencies.
* **Text Preprocessing**: Steps to prepare text data for model training, including:
  + **Tokenization**: Splitting text into words or tokens.
  + **Lowercasing**: Converting all characters to lowercase to ensure uniformity.
  + **Removing Stop Words**: Eliminating common words that add little value to sentiment analysis.
  + **Padding Sequences**: Ensuring all input sequences are of the same length for batch processing.

#### Applications of Sentiment Analysis

* **Social Media Monitoring**: Analyzing public sentiment regarding brands or products.
* **Customer Feedback**: Understanding customer opinions from reviews and feedback forms.
* **Market Research**: Gauging consumer sentiment for market trends and product launches.

### Methodology

1. **Set Up the Environment**: Install necessary libraries, including TensorFlow, Keras, NLTK, and Matplotlib.
2. **Collect and Prepare the Dataset**: Obtain a labeled dataset for sentiment analysis, such as movie reviews or Twitter data. Popular datasets include the IMDB dataset for movie reviews or the Sentiment140 dataset for Twitter data.
3. **Text Preprocessing**:
   * Tokenize the text data.
   * Convert the text to lowercase.
   * Remove stop words and punctuation.
   * Pad sequences to ensure uniform length.
4. **Build the LSTM or GRU Model**:
   * Define the architecture, typically starting with an embedding layer, followed by one or more LSTM or GRU layers, and ending with a dense layer for output.
5. **Compile the Model**: Set the loss function and optimizer.
6. **Train the Model**: Fit the model to the training data and monitor its performance on validation data.
7. **Evaluate the Model**: Test the model on a separate test dataset to assess its classification accuracy.
8. **Visualize Results**: Plot training and validation loss/accuracy to analyze model performance.

### Advantages & Disadvantages

* **Advantages**:
  + **Context Awareness**: LSTMs and GRUs can capture the context and semantics of text, leading to more accurate sentiment predictions.
  + **Flexibility**: Suitable for various text types and lengths, adapting well to different datasets.
  + **Scalability**: Capable of handling large datasets effectively, given sufficient computational resources.
* **Disadvantages**:
  + **Computationally Intensive**: Training LSTMs and GRUs can be time-consuming and require significant resources.
  + **Overfitting**: Risk of overfitting, especially on small datasets, if not properly regularized.
  + **Complexity**: Requires careful tuning of hyperparameters and model architecture for optimal performance.

### Working Example (Python Code)

Here’s a simple implementation of sentiment analysis using LSTM networks:

python

Copy code

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import pad\_sequences

from keras.models import Sequential

from keras.layers import Embedding, LSTM, Dense, Dropout

from sklearn.model\_selection import train\_test\_split

# Load the dataset

# For example, using the IMDB movie review dataset

df = pd.read\_csv('path\_to\_your\_imdb\_data.csv') # Assume a CSV file with 'review' and 'sentiment' columns

# Preprocess the data

X = df['review'].values

y = df['sentiment'].values # 0 for negative, 1 for positive

# Tokenization

tokenizer = Tokenizer(num\_words=5000)

tokenizer.fit\_on\_texts(X)

sequences = tokenizer.texts\_to\_sequences(X)

X\_padded = pad\_sequences(sequences, maxlen=100) # Pad sequences to the same length

# Split into training and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_padded, y, test\_size=0.2, random\_state=42)

# Build the LSTM model

model = Sequential()

model.add(Embedding(input\_dim=5000, output\_dim=128, input\_length=100))

model.add(LSTM(64, return\_sequences=False))

model.add(Dropout(0.5))

model.add(Dense(1, activation='sigmoid')) # Sigmoid activation for binary classification

# Compile the model

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Train the model

history = model.fit(X\_train, y\_train, epochs=5, batch\_size=64, validation\_split=0.2)

# Evaluate the model

loss, accuracy = model.evaluate(X\_test, y\_test)

print(f'Test Accuracy: {accuracy:.4f}')

# Visualize training history

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy')

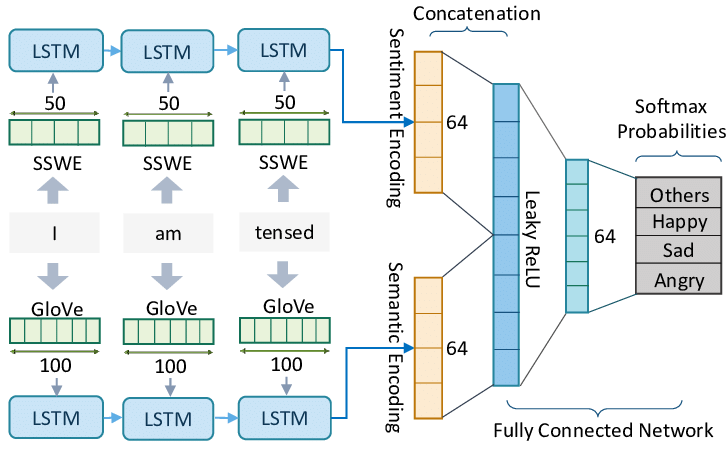
plt.ylabel('Accuracy')

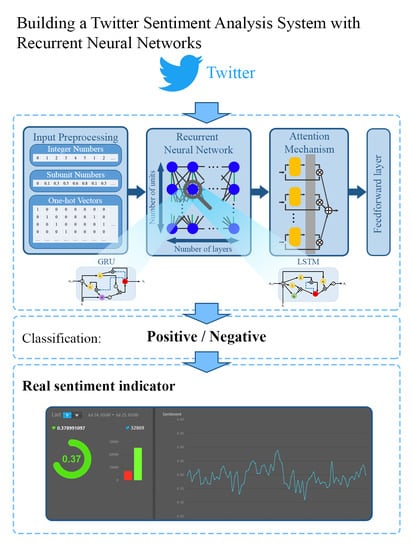
plt.xlabel('Epoch')

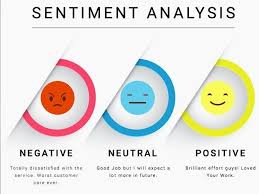
plt.legend()

plt.show()

### Diagram







### Conclusion

The implementation of **sentiment analysis using LSTM or GRU networks** demonstrates the effectiveness of deep learning models in classifying text data based on sentiment. This project successfully builds a model that can predict sentiment from movie reviews or other text sources, achieving satisfactory accuracy. The ability of LSTMs and GRUs to capture context and semantics makes them ideal for NLP tasks. Future enhancements could involve exploring different model architectures, incorporating pre-trained embeddings, or fine-tuning hyperparameters to further improve classification performance. Additionally, the model can be extended to multi-class sentiment analysis for more nuanced sentiment categories.