

**PAI LAB**

**Task # 9**

**Name:**

Saad Waleed

**Roll No:**

073

**Submitted To:**

Sir Rasik

**Date:**

5 April 2025

**Subject:**

PAI

**INTRO**  
This report details the implementation of a binary sentiment classifier for Foodpanda customer reviews using a Simple RNN. Reviews are labeled positive if their average rating exceeds 3.5, otherwise negative. After data loading, cleaning, tokenization, and padding, the model is trained and evaluated—with a sample prediction yielding a positive sentiment probability of **0.92**.

**TOOLS & TECHNOLOGY**

**Python 3.8+**

**Pandas**: data loading & cleaning

**NumPy**: numerical operations

**scikit-learn**:

train\_test\_split for stratified dataset splits

**TensorFlow / Keras**:

Tokenizer, pad\_sequences for text preprocessing

Sequential, Embedding, SimpleRNN, Dense for model definition, training, and evaluation

**Regular Expressions (re)**: text normalization

**CODE EXPLANATION**

# Import Libraries

import pandas as pd

import re

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

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, SimpleRNN, Dense

***Description:*** Load all required packages for data handling, preprocessing, model construction, and evaluation.

# Load & Clean Data

df = pd.read\_csv('foodpanda.csv').dropna()

df['text'] = (

df['Review']

.str.lower()

.replace(r'[^a-z0-9\s]', '', regex=True)

)

df['label'] = df['Avg Rating'].apply(lambda r: 1 if r > 3.5 else 0)

***Description****:* Reads the raw CSV, removes incomplete records, cleans and normalizes the review text, and generates the target labels for sentiment.

# Tokenization & Padding

VOCAB\_SIZE = 5000

MAX\_LEN = 200

tokenizer = Tokenizer(num\_words=VOCAB\_SIZE)

tokenizer.fit\_on\_texts(df['text'])

sequences = tokenizer.texts\_to\_sequences(df['text'])

X = pad\_sequences(sequences, maxlen=MAX\_LEN)

y = df['label'].values

***Description:*** Limits the vocabulary to the top 5,000 words, tokenizes each review into integer sequences, and applies pre-padding/truncation to ensure all inputs are length 200.

# Train / Validation / Test Split

X\_train, X\_temp, y\_train, y\_temp = train\_test\_split(

X, y,

test\_size=0.20,

random\_state=42,

stratify=y

)

X\_val, X\_test, y\_val, y\_test = train\_test\_split(

X\_temp, y\_temp,

test\_size=0.50,

random\_state=42,

stratify=y\_temp

)

***Description:*** Performs stratified splitting to maintain the same positive/negative ratio across training, validation, and test sets.

# Model Definition

model = Sequential([

Embedding(input\_dim=VOCAB\_SIZE, output\_dim=16, input\_length=MAX\_LEN),

SimpleRNN(64, activation='tanh'),

Dense(1, activation='sigmoid')

])

model.compile(

loss='binary\_crossentropy',

optimizer='adam',

metrics=['accuracy']

)

***Description:***

* **Embedding layer**: transforms each token into a 16-dimensional vector.
* **SimpleRNN**: processes the sequence and outputs a 64-unit hidden state.
* **Dense sigmoid**: produces a probability for the positive class.

# Training

history = model.fit(

X\_train, y\_train,

epochs=5,

batch\_size=32,

validation\_data=(X\_val, y\_val),

verbose=1

)

***Description:*** Trains the model for 5 epochs with batch size 32, monitoring validation loss and accuracy after each epoch.

# Evaluation

test\_loss, test\_acc = model.evaluate(X\_test, y\_test, verbose=0)

print(f"Test accuracy: {test\_acc:.2f}")

***Description:*** Evaluates the trained model on the held-out test set and reports overall accuracy.

# Prediction Function & Sample

def predict\_sentiment(text: str) -> float:

clean = re.sub(r'[^a-z0-9\s]', '', text.lower())

seq = tokenizer.texts\_to\_sequences([clean])

pad = pad\_sequences(seq, maxlen=MAX\_LEN)

prob = model.predict(pad, verbose=0)[0][0]

return prob

probability = predict\_sentiment("The food was great.")

print(f"Predicted positive probability: {probability:.2f}")

*Description:* Provides a utility to preprocess any raw review string, make a sentiment prediction, and return the predicted probability.

**CONCLUSION**  
The Simple RNN model achieves strong performance for binary sentiment classification on Foodpanda reviews, with a sample prediction probability of **0.92** for “The food was great.” To further enhance accuracy and robustness, consider the following next steps:

* **Switch to LSTM or GRU** layers to better capture long-range dependencies.
* **Use pre-trained embeddings** (e.g., GloVe, fastText) for richer lexical representations.
* **Bidirectional RNNs** or **CNN-RNN hybrids** for improved feature extraction.

**Hyperparameter tuning**: experiment with embedding dimensions, RNN units, learning rates, and dropout regularization.