

Configuration Manual

MSc Research Project Artificial Intelligence for Business

> Usman Ghani Student ID: x22186514

School of Computing National College of Ireland

Supervisor: VICTOR DEL ROSAL

1. Introduction

Our primary goal is to provide a cutting-edge sentiment analysis framework tailored specifically for FlickCard, a platform for user-generated content. This system leverages advanced deep learning models, including LSTM networks, to accurately classify sentiments as positive, negative, or neutral. We created the notification alert system using the pushbullet for fast replies when we received negative reviews on our products from our clients. In this document, important code snippets are present that can be used to recreate the project code.

2. System Requirements

There are two methods to execute and set up our code. The first method involves installing the Anaconda Navigator on our local machine. Alternatively, if we prefer not to install the Anaconda Navigator locally, we can utilize Google Colab. Google Colab is an online service that offers computational power and fast, free services for running our machine learning tasks. In our case, we use the google colab.

2.1 System Configuration

Google Colab setup

• Open Google and search for Google Colab, then click on the first link in the search results.

G google colab - Google Search × +	~	-
$\leftarrow \rightarrow \mathbf{C}$ (a google.com/search?q=google+colab&rlz=1C1GGRV_enPK1053PK1053&oq=goog&gs_lcrp=EgZjaHJvbWUq8ggCEEUYOzIPCAAQRRg7GIMBGLEDGIAEMg	È	☆
Google google colab X I C Q	()	***
Q All Videos Books News More Tools		SafeS
About 53,000,000 results (0.18 seconds)		
Google Research https://research.google.com > colaboratory Https://research.google.com > colaboratory Welcome To Colaboratory With Colab you can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses numpy to generate some random data Google Colab Notebook · Colab Pro · Cancel your subscription · Local runtimes colab.google https://colab.google itsp://colab.google Colab is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs.		
G Google Research https://research.google.com > colaboratory > faq : Google Colab More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources		

• After clicking on the URL, you will be directed to Google Colab. Here, locate the "File" option, and upon clicking, you will find the "New Notebook" option. Clicking on it will open a new notebook where you can write and execute your desired code.



• As we are utilizing online services, loading the dataset directly from our local machine isn't possible. Therefore, we need to upload our dataset. To do this, navigate to the "File" option, then select "Upload" and choose the data from your local folder to upload it to the drive.



a open				
\leftarrow \rightarrow \checkmark \uparrow 📜 « Compressed » archive_1	✓ U		🖻 🕁 🔇	
Organize • New folder	III • 🔟 🕐	vnh 🛧		
SThis PC Name	Date modified	ynd A		Comment 🛛 🎝 Share
3D Objects Image: Solution of the second	2/3/2023 9:21 AM			RAM
Desktop				V 14 Disk
Documents				1 4 🕫 🗖 🖌
Downloads				
J Music				
E Pictures				
🚆 Videos				
Local Disk (C:)				
New Volume (D:)				
🧼 New Volume (E:)				
✓ <	>	c		
File name: Dataset-SA	 All Files 	plt		
		loud		
	Open Cancei	classification_report		
	<pre>trom sklearn.metrics import import pickle</pre>	ť confusion_matrix		
	from sklearn.metrics import	t accuracy score		
		~_		
<>				
	 Load the dataset 			
Disk 51.27 GB available	[] # Load your dataset into a	pandas DataFrame		
	 Connected to Python 3 Google 	Compute Engine backend (GPU)		

3. Environment Setup

Google Colab typically comes pre-installed with many commonly used libraries. To use them, you just need to import them into your notebook. If any libraries are not already installed, you can easily install them using the pip command. In our specific case, we import the necessary libraries required for our work.

- Pandas
- Numpy
- Seaborn
- Spacy
- Matplotlib
- Wordcloud
- Tensorflow
- Sklearn
- Pickle

4. Implementation

4.1 Data Collection

• The dataset was obtained from Kaggle, and it's necessary to unzip the downloaded dataset before utilizing it. You can find the link to the dataset provided below.

https://www.kaggle.com/datasets/niraliivaghani/flipkart-product-customerreviews-dataset • As discussed in the earlier section on setting up Google Colab, we loaded the dataset onto the drive. Now, we proceed to read the dataset using the pandas read command.

\leftarrow	\rightarrow C	9	colab.research.google.com/drive/1iHv5AeZymG37_f38x	kGXTAz-1ZBdwA	EZ#scro	ollTo=WVlYyqu37Q	хк 🖻 🖈 🔇 🌐 🔁 🕝	🗈 🚺 🛛	• *	
C	File	Fli e E	pkart Product Reviews Sentiment Analysis and dit View Insert Runtime Tools Help <u>All changes sa</u>	d perfrom Act	ion.ip	ynb 🛧	E Com	ment 🖁	Share	٥
≣	+ C	ode	+ Text					Reco	nnect G	PU 🔻
Q { <i>x</i> }	C	•	<pre># Load your dataset into a pandas DataFrame df = pd.read_csv("/content/Dataset-SA.csv") df.head()</pre>							
Сī	Ξ	Ð	product_name	product_price	Rate	Review	Summary	Sentiment		
			Candes 12 L Room/Personal Air Cooler?????(Whi	3999	5	super!	great cooler excellent air flow and for this p	positive		
			1 Candes 12 L Room/Personal Air Cooler?????(Whi	3999	5	awesome	best budget 2 fit cooler nice cooling	positive		
		:	2 Candes 12 L Room/Personal Air Cooler?????(Whi	3999	3	fair	the quality is good but the power of air is de	positive		
		1	3 Candes 12 L Room/Personal Air Cooler?????(Whi	3999	1	useless product	very bad product its a only a fan	negative		
			4 Candes 12 L Room/Personal Air Cooler?????(Whi	3999	3	fair	ok ok product	neutral		
<>	~	da	ata preprocessing and EDA(Expl	oratory d	ata	analysis)				
>_			df.shape							

4.2 Data Preprocessing

• Check and remove missing values using the following code

```
## Check Missing values in our dataset
[]
      df.isnull().sum()
    product_name
                         0
                        ø
    product_price
    Rate
                        0
    Review
                    24664
    Summary
                     11
    Sentiment
                        0
    dtype: int64
```

- [] df.dropna(inplace=True)
 - Data visualization is done using matplotlib, seaborn
 - Implement a comprehensive cleaning process on the "Summary" column:
 - \succ Lowercase the text.
 - Tokenize using SpaCy.
 - Remove stop words.
 - > Apply lemmatization.
 - Save cleaned text in a new "cleaned_text" column.

```
# function for text cleaning
def clean_text(text_column):
   cleaned_texts = []
    for text in text_column:
       # Step 1: Text Lowercasing
       text = text.lower()
       # Step 2: Tokenization
       doc = nlp(text)
       tokens = [token.text for token in doc]
       # Step 3: Stop Word Removal
       filtered_tokens = [token for token in tokens if not nlp.vocab[token].is_stop]
       # Step 4: Lemmatization
       lemmatized_tokens = [token.lemma_ for token in nlp(" ".join(filtered_tokens))]
       # Step 5: Reconstruct the cleaned text
       cleaned_text = " ".join(lemmatized_tokens)
       cleaned_texts.append(cleaned_text)
   return cleaned texts
```

Clean the 'Summary review column' and create a new 'cleaned_text' column
df["cleaned_text"] = clean_text(df["Summary"])

4.3 Data Splitting

- Map sentiment labels to numerical values for LSTM model training.
- Split the dataset into input (X) and output (y) for model development
- Utilize scikit-learn's train_test_split function to divide the dataset into training and testing sets for robust model evaluation.

Create our independent and dependent columns

```
# Define a dictionary to map sentiment labels to numerical values
sentiment_mapping = {"negative": 0, "neutral": 1, "positive": 2}
# Replace values in the 'Sentiment' column using the dictionary
df['Sentiment'] = df['Sentiment'].map(sentiment_mapping)
```

X=df['cleaned_text']
y=df['Sentiment']

Splitting the Data:

```
#Split dataset into training and testing sets for model training and evaluation.
from sklearn.model_selection import train_test_split
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.2, random_state=42)
```

4.4 Tokenization and Padding

• Tokenize and pad the text data using Keras Tokenizer and pad_sequences to maintain consistent input lengths for the LSTM model.

```
#Tokenize the text and pad sequences to ensure consistent input lengths for the LSTM model.
# Create a tokenizer and fit it on the training data
tokenizer = Tokenizer(num_words=5000) # You can adjust the vocabulary size as needed
tokenizer.fit_on_texts(X_train)
# Save the tokenizer to a file
with open('tokenizer.pkl', 'wb') as tokenizer_file:
| pickle.dump(tokenizer, tokenizer_file)
# Convert text to sequences and pad them
X_train_sequences = tokenizer.texts_to_sequences(X_train)
X_test_sequences = tokenizer.texts_to_sequences(X_test)
max_sequence_length = 100 # Adjust the sequence length as needed
X_train_padded = pad_sequences(X_test_sequences, maxlen=max_sequence_length)
X_test_padded = pad_sequences(X_test_sequences, maxlen=max_sequence_length)
```

4.5 Model Building

• Train the LSTM model on the prepared dataset, adjusting key parameters based on validation results.

```
model = Sequential()
# Embedding layer: Maps words to dense vectors
model.add(Embedding(input_dim=5000, output_dim=32, input_length=max_sequence_length))
# LSTM layer: Processes sequential data
model.add(LSTM(100, return_sequences=True)) # Use return_sequences=True to stack LSTM layers
model.add(LSTM(75, return_sequences=True))
model.add(LSTM(50))
# Dense layers: Fully connected layers for learning complex patterns
model.add(Denpout(0.3)) # Dropout layer to reduce overfitting
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
# Output laver: Use softmax for multi-class classification
model.add(Dense(3, activation='softmax')) # 3 classes: negative, neutral, positive
# Compile the model
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# Train the model
model.fit(X_train_padded, y_train,
         validation_data=(X_test_padded, y_test),
          epochs=15,
         batch_size=128)
```

5. Model Evaluation

- Evaluate the trained LSTM model on the test set, considering metrics such as accuracy, loss, size, and latency.
- Generate a confusion matrix for a detailed performance assessment.

• Save the trained model for later use

```
# Save the LSTM model to a file
lstm_model_filename = 'saved_lstm_model.h5'
model.save(lstm_model_filename)
```

6. Negative Feedback Handling

6.1 Load The model

For negative feedback, we made a notification system. We save the model in the previous so we load that on the new notebook and also load the tokenizer pickle file.

\leftrightarrow \rightarrow C $($ colab.research.g	google.com/drive/1cS	ZoDyqYCe1M9FiFgTxxLp8vpPWrVBc5	Q 🖻 ☆
CO Action perfrom note File Edit View Insert Run	book.ipynb 🕁 time Tools Help Las	st saved at 8:01 PM	
i⊟ Files	- × + Code	+ Text	
Q 🚹 🖬 🔯	~ 0	<pre>!pip install pushbullet.py</pre>	
<pre>{x} {x}</pre>	~ [8]	from pushbullet import Pushbullet from tensorflow.keras.models import load_model import pickle from tensorflow.keras.preprocessing.sequence import import numpy as np	pad_sequences
	✓ [4]	<pre># Load the saved LSTM model loaded_lstm_model = load_model('<u>/content/saved_lstm</u></pre>	1_model.h5')
	✓ [6]	<pre>#load the tokenizer with open('tokenizer.pkl', 'rb') as tokenizer_file:</pre>	
	✓ [9]	<pre>def get_sentiment_prediction(text, model): # Tokenize and pad the input text # In another file or session,</pre>	
		text_sequence = loaded_tokenizer.texts_to_seque	<pre>ences([text]) ences([text])</pre>
		text_padded = pad_sequences(text_sequence, maxi	en=100)
Disk 80.78	GB available	<pre># Make sentiment prediction using the trained m prediction = model.predict(text_padded)</pre>	.ode1
		 Connected to Python 3 Google Compute Engir 	e backend
🛨 🔎 Type here to search	📃 🏂 🥯 📜 🗄 İ	- <u>e</u> = = <u>-</u> <u>-</u>	

6.2 Model Prediction

• Made a function in which we will pass the message it will return the prediction.



6.3 Notification System

- Implement a notification system for negative feedback using the Pushbullet library.
- Send notifications based on the model's predictions, enhancing user engagement and allowing timely responses to negative sentiments.

C		# Example usage:
0	-	# Replace 'your text here' with the actual text you want to predict
	•	<pre>input_text = input("Please enter the feedback:\n")</pre>
		# Call the function to get sentiment prediction
		<pre>predicted_sentiment = get_sentiment_prediction(input_text, loaded_lstm_model)</pre>
		# Define your notification message based on the prediction
		if predicted_sentiment == 'Negative':
		# Replace 'YOUR API KEY' with your actual Pushbullet API key
		<pre>pb = Pushbullet('o.9TzXjIdLL0vxGdFqaiODkvwZ8xPFH1t2')</pre>
		# Send a notification
		<pre>push = pb.push_note("Negative Sentiment Detected", "The sentiment analysis model has detected a negative sentiment. Please check the results.") print("Notification sent:", push)</pre>
		# Print the prediction and notification details
		print("Prediction:", predicted_sentiment)
••	• P	lease enter the feedback: