SENTIMENT ANALYSIS FOR MARKETING

Team Members:

Arunkumar P – 211521243023

Balamurugan K – 211521243030

Jaswanth Kumar S – 211521243072

Kathir Nilavan V – 211521243087

Kishore S – 211521243091

Phase 5 - Project Documentation:

Problem Definition:

This project aims to perform sentiment analysis on customer feedback expressed through tweets on Twitter regarding US airlines, with the goal of gaining actionable insights for marketing strategies. The primary objective is to accurately classify tweets into positive, negative, or neutral sentiments and extract meaningful topics to inform and guide critical business decisions within the US airline industry.

Design Thinking:

1.Data Collection:

The data utilized for the sentiment analysis project has been sourced from Kaggle, a renowned platform for datasets and data science resources.

Dataset Link: **https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment**

Github Link: https://github.com/Kishore-s-19/Sentiment-analysis-for-marketing.git

2. Data Preprocessing:

Effective data preprocessing is essential for ensuring that text data is clean, consistent and ready for sentiment analysis, which ultimately helps in extracting meaningful insights for marketing decisions.

- **Text Cleaning:**

  Convert text to lowercase for uniformity. Remove special characters, symbols and punctuation. Eliminate URLs and HTML tags.

- **Tokenization:**

  Break text into individual words or tokens.

- **Stopword Removal:**

  Remove common stopwords (e.g., "the", "and", "is").

- **Lemmatization or Stemming:**

  Reduce words to their base or root forms.

- **Handling Contractions and Negations:**

  Expand contractions (e.g., "can't" to "cannot"). Recognize and handle negations (e.g., "not good" vs. "good").

- **Removing Duplicates and Noise:**

  Identify and remove duplicate or near-duplicate texts. Filter out noisy or irrelevant data points.

- **Handling Missing Data:**

  Address missing data through imputation or sample removal.

- **Text Length Normalization:**

  Normalize text length by padding or truncating.

- **Exploratory Data Analysis (EDA):**

  Perform EDA to identify patterns and trends in the data.

- **Text Vectorization:**

  Convert preprocessed text into numerical vectors suitable for machine learning models.

3. Sentiment Analysis Techniques:

The choice of the best NLP technique should align with our project's objectives, available data and resource constraints. It's often beneficial to experiment with multiple techniques and evaluate their performance on a validation dataset to determine which one provide the most meaningful insights for guiding our marketing decisions. Here are some NLP techniques that can be used:

- **Deep Learning Models:**

  Recurrent Neural Networks (RNNs): Suitable for sequence data, they can capture context in text. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are popular RNN variants.

- **Bag of Words (BoW) and TF-IDF:**

  BoW represents text as a vector of word frequencies. TF-IDF (Term Frequency - Inverse Document Frequency) assigns weights to words based on their importance in a document relative to a corpus.

- **Transformer Models:**

  Transformer models have achieved state-of-the-art performance in various NLP tasks, including sentiment analysis. They can capture complex language patterns and context effectively. They are suitable for large datasets and high computational resources. They are recommended if we aim for top-tier accuracy and have access to pre-trained models. Ex: BERT, GPT-3

- **Word Embeddings:**

  Word embeddings are useful for capturing semantic relationships between words. They provide meaningful vector representations of words but might not capture context as well as transformer models. Ex: Word2Vec, GloVe

4. Feature Extraction:

Feature extraction is a critical step in sentiment analysis. It involves converting raw text data into numerical features that machine learning models can understand and use for analysis. We can convert text data into numerical features using methods like BoW, TF-IDF, or word embeddings.

5. Visualization:

Visualization plays a crucial role in sentiment analysis as it helps to make the analysis results more interpretable and actionable. It not only helps in understanding the data but also in conveying insights to stakeholders effectively. Visualization techniques such as matplotlib, seaborn can be used.

6. Insights Generation:

- **Aspect-Based Sentiment Insights:**

  Identification of specific aspects of the airline experience (e.g., customer service, pricing, seat comfort) that are associated with positive and negative sentiments. Understanding which aspects drive customer satisfaction and which require improvement.

- **Competitor Analysis:**

  Comparative analysis of sentiment scores among different US airlines. Insights into which airlines consistently receive positive feedback and which face challenges in satisfying customers.

- **Temporal Trends:**

  Identification of temporal trends in sentiment, such as seasonal fluctuations or changes over months and years. Correlation of sentiment trends with specific events or marketing campaigns.

- **Emotion Analysis:**

  Insights into the prevalent emotions expressed by customers in their feedback (e.g., happiness, frustration, gratitude).How emotions relate to specific aspects of the airline experience.

- **Geographic Patterns:**

  Regional analysis of sentiment to uncover geographic patterns in customer satisfaction. Understanding how sentiment varies across different locations and regions.

- **Response Effectiveness:**

  Evaluation of how airlines' responses to customer feedback on Twitter impact sentiment. Identification of response strategies that lead to improved customer satisfaction.

- **Complaint Analysis:**

  Analysis of the most common complaints and pain points mentioned by customers. Prioritization of issues that require immediate attention and resolution.

- **Sentiment by Customer Segment:**

  Segmentation of customers based on demographics or behaviors (e.g., frequent flyers, business travelers). Insights into how different customer segments perceive the airline experience.

- **Visualization Insights:**

  Insights from visualizations such as charts, graphs, and word clouds that highlight sentiment distribution and key terms. Visual representations of sentiment trends and patterns.

- **Recommendations:**

  Actionable recommendations for improving customer satisfaction and addressing pain points based on sentiment analysis findings. Prioritized strategies to enhance the overall customer experience.

These potential insights can provide a comprehensive understanding of customer sentiments and preferences related to US airlines. They serve as a valuable resource for data-driven decision-making, marketing strategies, customer service improvements, and business decisions aimed at enhancing customer satisfaction and competitiveness in the airline industry.

Innovation:

The innovation phase of our project represents a critical juncture in advancing our approach to sentiment analysis and using it to inform and enhance marketing efforts for US airlines. In this document, we will outline the strategies and methodologies to put our design thinking into innovation, focusing on the incorporation of advanced techniques such as LSTM (Long Short-Term Memory) from deep learning, Bag of Words, Transformer models, and Word Embeddings. These techniques will empower us to refine our sentiment analysis and gain deeper insights from tweets regarding US airlines.

1. Integration of LSTM for Enhanced Sentiment Analysis:

In our project to analyze sentiment in tweets regarding US airlines and improve marketing efforts, the integration of LSTM (Long Short-Term Memory) represents a powerful enhancement to our sentiment analysis pipeline. LSTM is a type of recurrent neural network (RNN) architecture known for its ability to capture sequential dependencies and long-range context in data. Here's a detailed explanation of how LSTM can be integrated and the benefits it brings to sentiment analysis.

- **Preprocess the data:** This involves cleaning the tweets, removing stop words, and converting the tweets to a numerical representation. One way to do this is to use a word embedding model, which converts each word to a vector of real numbers.

- **Train the LSTM model:** Once the data is preprocessed, you can train your LSTM model on a labeled dataset of tweets. The labeled dataset should contain tweets that have been labeled with their sentiment, such as positive, negative, or neutral.

- **Predict the sentiment of new tweets:** Once the LSTM model is trained, you can use it to predict the sentiment of new tweets. This involves feeding the tweet into the model and getting the output. The output of the model will be a probability distribution over the different sentiment classes (positive, negative, neutral).

Benefits of using LSTM for sentiment analysis:

- **Accuracy:** LSTM models have been shown to be more accurate than other sentiment analysis techniques, such as bag of words. This is because LSTM models are able to learn long-range dependencies in text, which is important for sentiment analysis.

- **Robustness:** LSTM models are more robust to noise and ambiguity in text than other sentiment analysis techniques. This is because LSTM models are able to learn the context of words in a sentence, which helps them to identify the correct sentiment.

- **Scalability:** LSTM models can be trained on large datasets of text, which allows them to learn more complex relationships between words and phrases. This can lead to more accurate sentiment analysis predictions.

Overall incorporating LSTM into our sentiment analysis pipeline brings several advantages, including improved contextual understanding, better handling of long-term dependencies, enhanced accuracy, flexibility with variable-length texts, and the potential for transfer learning. It enables our model to make more nuanced and accurate sentiment predictions, which can significantly contribute to our project's goal of enhancing marketing efforts for US airlines based on customer feedback in tweets.

2. Exploring Transformer Models for Sentiment Analysis:

In our project to analyze sentiment in tweets regarding US airlines and improve marketing efforts, the exploration of Transformer models represents a cutting-edge approach to enhance our sentiment analysis capabilities. Transformer models, such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer) have revolutionized natural language processing (NLP) tasks due to their ability to capture contextual information and semantic nuances in language data. Here's a detailed explanation of how Transformer models can be explored and the benefits they offer for sentiment analysis.

- **Bidirectional Context:** Transformer models are designed to capture bidirectional context, meaning they consider both preceding and following words in a text when processing each word/token. This is in contrast to traditional models that read text sequentially in one direction.

- **Self-Attention Mechanism:** Transformers utilize a self-attention mechanism that assigns different levels of importance to different parts of the input text, allowing them to focus on relevant context for each word/token.

- **Pre-trained Models:** Transformer models are often pre-trained on massive text corpora, which gives them a deep understanding of language semantics and grammar. These pre-trained models can then be fine-tuned for specific NLP tasks like sentiment analysis.

Benefits of using transformer models for sentiment analysis:

- **Accuracy:** Transformer models have been shown to be more accurate than other sentiment analysis techniques, such as bag of words and Naive Bayes. This is because transformer models are able to learn long-range dependencies in text, which is important for sentiment analysis.

- **Robustness:** Transformer models are more robust to noise and ambiguity in text than other sentiment analysis techniques. This is because transformer models are able to learn the context of words in a sentence, which helps them to identify the correct sentiment.

- **Efficiency:** Transformer models are more efficient than other NLP models, such as RNNs. This is because transformer models do not require recurrent connections, which can be computationally expensive.

- **Scalability:** Transformer models can be trained on large datasets of text, which allows them to learn more complex relationships between words and phrases. This can lead to more accurate sentiment analysis predictions.

- **Multilingual Support:** Transformer models can be used for sentiment analysis in multiple languages, which is valuable for analyzing tweets in different languages.

The exploration of Transformer models offers several advantages for sentiment analysis in tweets. These models provide a deeper understanding of language context, semantics, and nuances, enabling more accurate and context-aware sentiment analysis. Their ability to handle ambiguity and support multiple languages, along with the benefits of transfer learning and few-shot learning, makes them valuable tools for improving marketing efforts based on customer feedback in tweets. Transformer models represent the state-of-the-art in NLP and can significantly contribute to the success of our sentiment analysis project.

3. **Leveraging Bag of Words (BoW) and Word Embeddings for Sentiment Analysis:**

These techniques are widely used in natural language processing (NLP) for various tasks, including sentiment analysis. BoW is a simple but effective technique for representing text. BoW models represent each tweet as a vector of word counts. The sentiment of a tweet is then predicted based on the distribution of words in the vector. Word embeddings are a more sophisticated way of representing text. Word embeddings are vectors of real numbers that capture the semantic and syntactic relationships between words. Word embeddings can be used to improve the performance of BoW and other sentiment analysis models.

Implementation of Bag of Words:

- **Tokenization:** The first step is to tokenize each tweet, breaking it down into individual words or tokens. You can use NLP libraries like NLTK or spaCy for this task.

- **Vocabulary Building:** Create a vocabulary by compiling a list of unique words (tokens) from all the tweets in your dataset. This vocabulary serves as the basis for feature representation.

- **Vectorization:** Transform each tweet into a numerical vector representation based on the vocabulary. In the BoW model, this is typically done by counting the frequency of each word in the tweet and mapping it to its corresponding index in the vocabulary. Alternatively, you can use binary encoding (1 for presence, 0 for absence) or term frequency-inverse document frequency (TF-IDF) weighting.

- **Feature Matrix:** As a result, you'll obtain a feature matrix where each row represents a tweet and each column represents a word in the vocabulary. The values in the matrix represent the frequency, binary presence or TF-IDF weight of each word in the corresponding tweet.

Benefits of using Bag of words:

- **Simplicity:** BoW is easy to understand and implement, making it accessible even to individuals with limited NLP expertise.

- **Interpretability:** The resulting BoW feature matrix is interpretable. It allows users to see which words are contributing to the sentiment analysis, aiding in understanding the sentiment classification process.

- **Efficiency:** BoW is computationally efficient, making it suitable for processing large datasets and real-time applications.

- **Customization:** BoW can be customized to fit specific project requirements. We can adjust it by considering word frequency, binary presence, or more advanced techniques like TF-IDF, tailoring it to your needs.

- **Scalability:** BoW scales well with the dataset size and is applicable to both binary (positive/negative) and multi-class sentiment classification tasks.

Implementation of Word Embeddings:

- **Pre-trained Word Embeddings:** Utilize pre-trained word embeddings models like Word2Vec, GloVe, or FastText. These models have been trained on vast text corpora and capture semantic relationships between words.

- **Word-to-Vector Mapping:** Convert each word in a tweet into its respective word vector from the pre-trained model. This mapping transforms words into high-dimensional numerical vectors.

- **Vector Aggregation:** To represent a tweet as a fixed-size vector, you can aggregate the word vectors using techniques like averaging (mean of word vectors), summation, or weighted summation based on TF-IDF scores.

Benefits of using Word Embeddings:

- **Semantic Understanding**: Word embeddings capture semantic relationships between words, enabling models to understand context, word meanings and associations among words.

- **Dimensionality Reduction:** Word embeddings reduce the dimensionality of text data, transforming words into numerical vectors with lower dimensions. This reduces computational complexity and minimizes the risk of overfitting.
- **Contextual Information:** Word embeddings consider the context in which words appear, allowing models to capture how words are used in different contexts. This is vital for handling nuances and connotations in sentiment analysis.

- **Generalization:** Pre-trained word embeddings, such as Word2Vec, GloVe, or FastText, generalize well across various NLP tasks and domains. They can be fine-tuned for specific sentiment analysis scenarios, saving time and resources.

- **Enhanced Performance:** Word embeddings often lead to improved sentiment analysis performance compared to simpler techniques like BoW. They capture the semantic and contextual information necessary for accurate sentiment classification.

In summary, while Bag of Words is a straightforward and interpretable technique, Word Embeddings offer a deeper understanding of language semantics and context. Choosing between these techniques depends on project requirements, dataset characteristics, and the level of semantic understanding needed for the specific sentiment analysis task. In many cases, a combination of both techniques can yield the best results, as they complement each other's strengths.

Performance:

For our project advanced techniques like LSTM and Transformer models (e.g., BERT, RoBERTa) are likely to outperform simpler techniques like Bag of Words (BoW) and basic Word Embeddings. These advanced methods have proven to achieve superior performance and accuracy in capturing the contextual nuances and semantics of natural language, making them highly suitable for sentiment analysis tasks in the complex and nuanced world of social media, including tweets.

Specifically LSTM with its sequential analysis capabilities, is an effective choice for understanding the context and sequential dependencies in tweets. It provides good accuracy and context-awareness for sentiment analysis.

In terms of better performance and accuracy, Transformer models, such as BERT and RoBERTa, are generally considered the top performers in NLP tasks. However, the choice between LSTM and Transformer models should consider factors like the availability of pre-trained models, computational resources, and project goals.

In summary, for our sentiment analysis project involving US airline tweets, leveraging Transformer models like BERT or RoBERTa would likely provide the highest level of performance and accuracy due to their advanced language understanding capabilities and context-awareness.

1 IMPORT MODULES

[3]:

```
pip install transformers
```

```
Collecting transformers
  Downloading transformers-4.34.1-py3-none-any.whl (7.7 MB)
                                                7.7/7.7 MB
73.2 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from
transformers) (3.12.4)
Collecting huggingface-hub<1.0,>=0.16.4 (from transformers) Downloading
  huggingface_hub-0.18.0-py3-none-any.whl (301 kB)
                                                302.0/302.0
kB 36.2 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (1.23.5)Requirement already satisfied:
packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (23.2) Requirement already
satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2023.6.3) Requirement already
satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.31.0)
Collecting tokenizers<0.15,>=0.14 (from transformers)
  Downloading
tokenizers-0.14.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.8 MB)
                                                3.8/3.8 MB
106.7 MB/s eta 0:00:00
Collecting safetensors>=0.3.1 (from transformers)
  Downloading
safetensors-0.4.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
                                                1.3/1.3 MB
91.5 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.27 in
```

```
/usr/local/lib/python3.10/dist-packages (from transformers) (4.66.1)Requirement already
satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub<1.0,>=0.16.4->transformers) (2023.6.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub<1.0,>=0.16.4->transformers) (4.5.0)
Collecting huggingface-hub<1.0,>=0.16.4 (from transformers) Downloading
    huggingface_hub-0.17.3-py3-none-any.whl (295 kB)
                                          295.0/295.0

kB 34.3 MB/s eta 0:00:00
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.3.1) Requirement already
satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)Requirement already
satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers)(2023.7.22)
Installing collected packages: safetensors, huggingface-hub, tokenizers,transformers
Successfully installed huggingface-hub-0.17.3 safetensors-0.4.0tokenizers-0.14.1
transformers-4.34.1
```

[4]:

```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.14.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from
tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)Requirement
already satisfied: flatbuffers>=23.5.26 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26) Requirement already
satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5.4)Requirement
already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from
tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.6)Requirement already
satisfied: ml-dtypes==0.2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from
tensorflow) (1.23.5)
```

Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0) Requirement already
satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.2)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3) Requirement already
satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from
tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.3.0)Requirement already
satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)Requirement already
satisfied: wrapt<1.15,>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1) Requirement already
satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.34.0)Requirement already
satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.59.0)Requirement already
satisfied: tensorboard<2.15,>=2.14 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.1) Requirement
already satisfied: tensorflow-estimator<2.15,>=2.14.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.0)Requirement already
satisfied: keras<2.15,>=2.14.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.14.0)Requirement already
satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow)(0.41.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.15,>=2.14->tensorflow) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.15,>=2.14->tensorflow) (1.0.0) Requirement
already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.15,>=2.14->tensorflow) (3.5) Requirement already
satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.15,>=2.14->tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.15,>=2.14->tensorflow) (0.7.2) Requirement
already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
tensorboard<2.15,>=2.14->tensorflow) (3.0.1)

Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google- auth<3,>=1.6.3->tensorboard<2.15,>=2.14->tensorflow) (5.3.2)Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google- auth<3,>=1.6.3->tensorboard<2.15,>=2.14->tensorflow) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist- packages (from google-auth<3,>=1.6.3->tensorboard<2.15,>=2.14->tensorflow) (4.9)Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth- oauthlib<1.1,>=0.5->tensorboard<2.15,>=2.14->tensorflow) (1.3.1)Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.15,>=2.14->tensorflow) (3.3.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist- packages (from requests<3,>=2.21.0->tensorboard<2.15,>=2.14->tensorflow) (3.4) Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.15,>=2.14->tensorflow) (2.0.7)Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from
requests<3,>=2.21.0->tensorboard<2.15,>=2.14->tensorflow) (2023.7.22) Requirement already satisfied: MarkupSafe>=2.1.1 in
[5]: /usr/local/lib/python3.10/dist-packages (from
werkzeug>=1.0.1->tensorboard<2.15,>=2.14->tensorflow) (2.1.3)Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in

[6]:

```python
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.feature_extraction.text import CountVectorizer
pip install keras
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.models import Sequential
```

```python
from transformers.models.bert.modeling_bert import BertModel,BertForMaskedLM

import matplotlib.pyplot as plt
import seaborn as sns
```

[8]:

```python
# Load your sentiment dataset (e.g., from a CSV file)
data = pd.read_csv("Tweets.csv")
```

DATA PREPROCESSING

[9]:

```python
texts = [[word.lower() for word in text.split()] for text in data]
```

[10]:

```python
data.head()
```

[10] :

	tweet_id	airline_sentiment	airline_sentiment_confidence	\
0	570306133677760513	neutral	1.0000	
1	570301130888122368	positive	0.3486	
2	570301083672813571	neutral	0.6837	
3	570301031407624196	negative	1.0000	
4	570300817074462722	negative	1.0000	

	negativereason	negativerea son_confidence	airline	\
0	NaN	NaN	Virgin America	
1	NaN	0.0000	Virgin America	
2	NaN	NaN	Virgin America	
3	Bad Flight	0.7033	Virgin America	
4	Can't Tell	1.0000	Virgin America	

	airline_sentiment_gold	name	negativereason_gold	retweet_count	\
0	NaN	cairdin	NaN	0	
1	NaN	jnardino	NaN	0	
2	NaN	yvonnalynn	NaN	0	
3	NaN	jnardino	NaN	0	
4	NaN	jnardino	NaN	0	

	text	tweet_coord	\
0	@VirginAmerica What @dhepburn said.	NaN	
1	@VirginAmerica plus you've added commercials t…	NaN	
2	@VirginAmerica I didn't today… Must mean I n…	NaN	
3	@VirginAmerica it's really aggressive to blast…	NaN	
4	@VirginAmerica and it's a really big bad thing…	NaN	

	tweet_created	tweet_location	user_timezone
0	2015-02-24 11:35:52 -0800	NaN	Eastern Time (US & Canada)
1	2015-02-24 11:15:59 -0800	NaN	Pacific Time (US & Canada)
2	2015-02-24 11:15:48 -0800	Lets Play	Central Time (US & Canada)
3	2015-02-24 11:15:36 -0800	NaN	Pacific Time (US & Canada)

```
4   2015-02-24 11:14:45 -0800                     NaN  Pacific Time (US & Canada)
```

[ ]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14640 entries, 0 to 14639 Data
columns (total 15 columns):
 #   Column                        Non-Null Count  Dtype
------  -----------                    --------------------------  ----------
 0    tweet_id                      14640 non-null   int64
 1    airline_sentiment             14640 non-null   object
 2    airline_sentiment_confidence  14640 non-null   float64
 3    negativereason                9178 non-null    object
 4    negativereason_confidence     10522 non-null   float64
 5    airline                       14640 non-null   object
 6    airline_sentiment_gold        40 non-null      object
 7    name                          14640 non-null   object
 8    negativereason_gold           32 non-null      object
 9    retweet_count                 14640 non-null   int64
 10   text                          14640 non-null   object
 11   tweet_coord                   1019 non-null    object
 12   tweet_created                 14640 non-null   object
 13   tweet_location                9907 non-null    object
 14   user_timezone                 9820 non-null    object
dtypes: float64(2), int64(2), object(11)
memory usage: 1.7+ MB
```

[ ]:

```
data.isnull().sum()
```

[ ]:

```
tweet_id                            0
airline_sentiment                   0
airline_sentiment_confidence        0
negativereason                   5462
negativereason_confidence        4118
airline                             0
airline_sentiment_gold          14600
name                                0
negativereason_gold             14608
retweet_count                       0
text                                0
tweet_coord                     13621
tweet_created                       0
tweet_location                   4733
user_timezone                    4820
dtype: int64
```

[ ]:

```
data.describe()
```

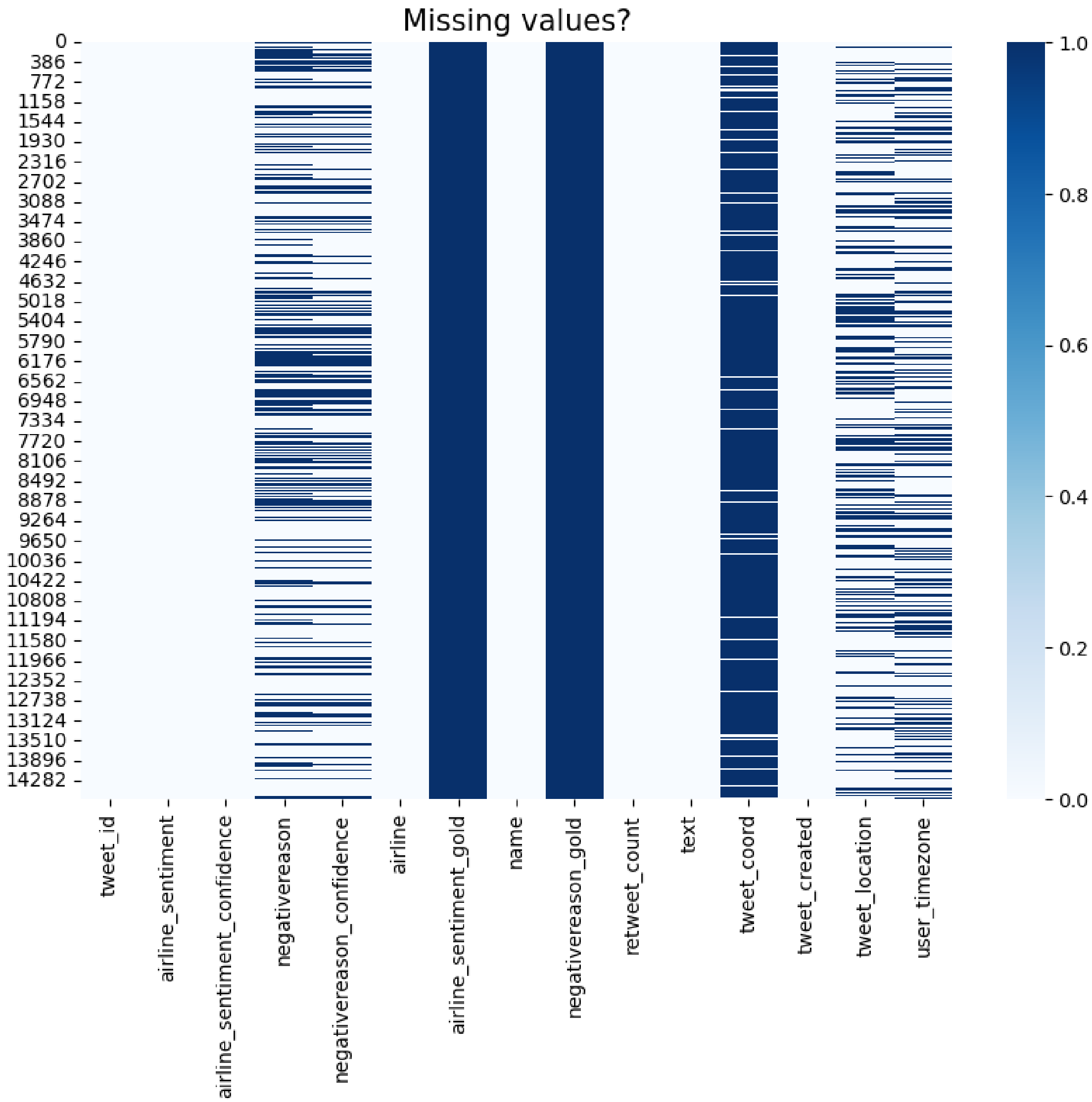
[ ]:

	tweet_id	airline_sentiment_confidence	negativereason_confidence \
count	1.464000e+04	14640.000000	10522.000000
mean	5.692184e+17	0.900169	0.638298
std	7.791112e+14	0.162830	0.330440
min	5.675883e+17	0.335000	0.000000
25%	5.685592e+17	0.692300	0.360600
50%	5.694779e+17	1.000000	0.670600
75%	5.698905e+17	1.000000	1.000000
max	5.703106e+17	1.000000	1.000000

	retweet_count
count	14640.000000
mean	0.082650
std	0.745778
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	44.000000

[ ]:

```
#Visualization of missing value using heatmap
plt.figure(figsize=(10,7))
sns.heatmap(data.isnull(), cmap = "Blues")
plt.title("Missing values?", fontsize = 15)
plt.show()
```



[ ]:
```
print("Percentage null or na values in data")
((data.isnull() | data.isna()).sum() * 100 / data.index.size).round(2)
```

Percentage null or na values in data

[ ]:

tweet_id	0.00
airline_sentiment	0.00
airline_sentiment_confidence	0.00
negativereason	37.31
negativereason_confidence	28.13
airline	0.00
airline_sentiment_gold	99.73
name	0.00

```
negativereason_gold            99.78
retweet_count                   0.00
text                            0.00
tweet_coord                    93.04
tweet_created                   0.00
tweet_location                 32.33
user_timezone                  32.92
dtype: float64
```

[ ]:
```python
data.drop(["tweet_coord", "airline_sentiment_gold", "negativereason_gold"],␣
 ↪axis=1, inplace=True)
```

[ ]:
```python
freq = data.groupby("negativereason").size()
```

[ ]:
```python
data.duplicated().sum()
```

[ ]: 39

[ ]:
```python
# Dropping duplicates
data.drop_duplicates(inplace = True)
```

[ ]:
```python
data.duplicated().sum()
```

[ ]: 0

[ ]:
```python
data.describe().T
```

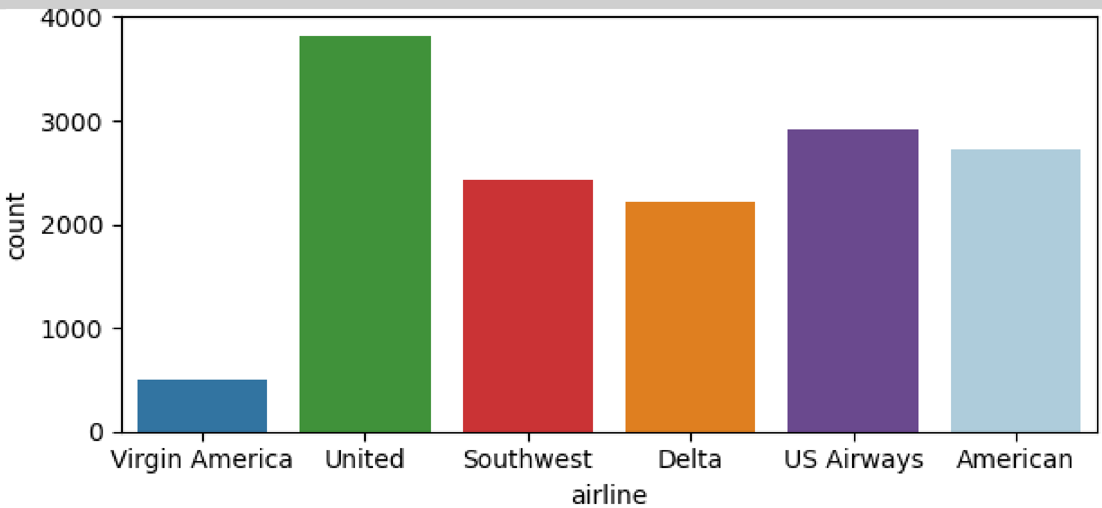
[ ]:
```
                                count          mean           std  \
tweet_id                      14601.0  5.692156e+17  7.782706e+14
airline_sentiment_confidence  14601.0  8.999022e-01  1.629654e-01
negativereason_confidence     10501.0  6.375749e-01  3.303735e-01
retweet_count                 14601.0  8.280255e-02  7.467231e-01

                                       min           25%           50%  \
tweet_id                      5.675883e+17  5.685581e+17  5.694720e+17
airline_sentiment_confidence  3.350000e-01  6.923000e-01  1.000000e+00
negativereason_confidence     0.000000e+00  3.605000e-01  6.705000e-01
retweet_count                 0.000000e+00  0.000000e+00  0.000000e+00

                                       75%           max
tweet_id                      5.698884e+17  5.703106e+17
airline_sentiment_confidence  1.000000e+00  1.000000e+00
negativereason_confidence     1.000000e+00  1.000000e+00
retweet_count                 0.000000e+00  4.400000e+01
```

2 EXPLORATORY DATA ANALYSIS

[ ]:
```
data.nunique()
```

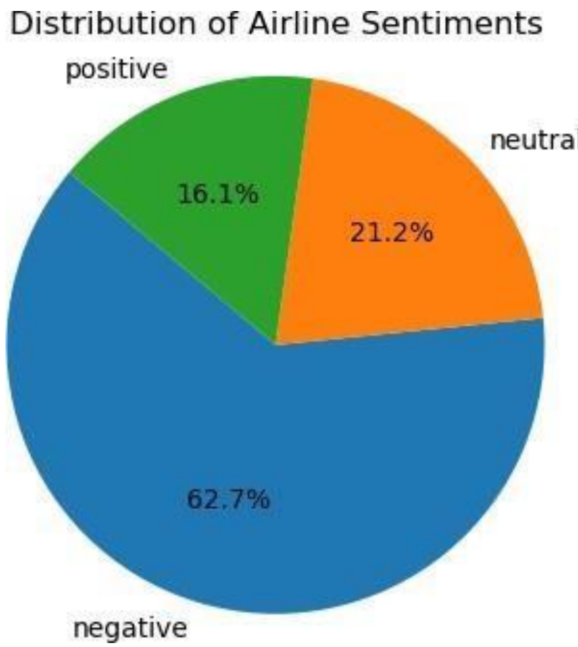
[ ]:
```
tweet_id                        14485
airline_sentiment                   3
airline_sentiment_confidence     1023
negativereason                     10
negativereason_confidence        1410
airline                             6
name                             7701
retweet_count                      18
text                            14427
tweet_created                   14247
tweet_location                   3081
user_timezone                      85
dtype: int64
```

[ ]:
```
# Checking the distribution of airlines
plt.figure(figsize=(7,3))
sns.countplot(data=data,x='airline', palette=['#1f78b4', '#33a02c', '#e31a1c',␣
↪'#ff7f00', '#6a3d9a', '#a6cee3'])
plt.show()
```

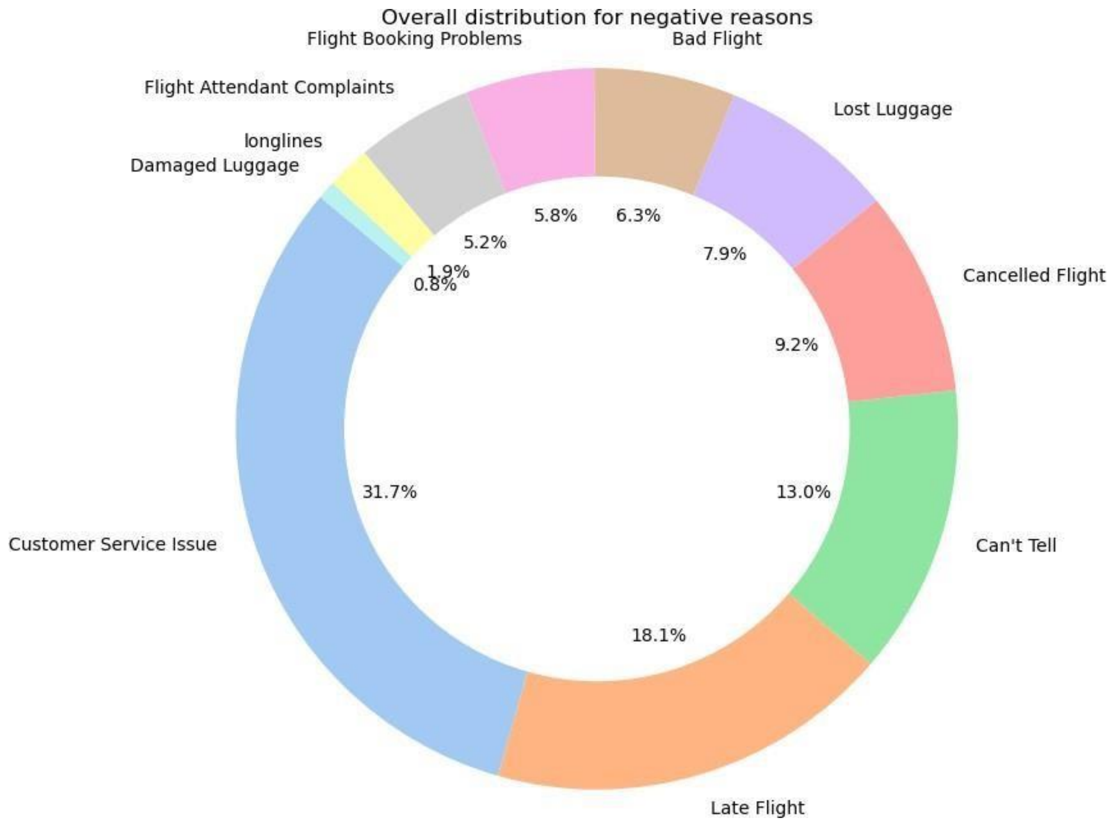


[ ]:
```
sentiment_counts = data['airline_sentiment'].value_counts()
plt.figure(figsize=(6, 4))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%',␣
↪startangle=140)
plt.title('Distribution of Airline Sentiments')
```

```
plt.axis("equal")  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```



[ ]:
```
# Calculate the value counts for each negative reason
value_counts = data["negativereason"].value_counts()

# Create a donut-like pie chart using matplotlib and seaborn
plt.figure(figsize=(8, 8))
labels = value_counts.index
values = value_counts.values
colors = sns.color_palette("pastel")[0:len(labels)]  # Use pastel colors for␣
↪the chart
plt.pie(values, labels=labels, colors=colors, autopct="%1.1f%%",␣
↪startangle=140, wedgeprops=dict(width=0.3))
plt.title("Overall distribution for negative reasons")
plt.axis("equal")  # Equal aspect ratio ensures the pie chart is drawn as a␣
↪circle.
plt.show()
```



[10]:

```
# Selec  only the necessary columns for sentiment analysis
data = data[['airline_sentiment', 'text']].copy()
data
```

[10]:

0	neutral	@VirginAmerica What @dhepburn said.
1	positive	@VirginAmerica plus you've added commercials t…
2	neutral	@VirginAmerica I didn't today… Must mean I n…
3	negative	@VirginAmerica it's really aggressive to blast…
4	negative	@VirginAmerica and it's a really big bad thing…
...	...	...
14635	positive	@AmericanAir thank you we got on a different f…
14636	negative	@AmericanAir leaving over 20 minutes Late Flig…
14637	neutral	@AmericanAir Please bring American Airlines to…
14638	negative	@AmericanAir you have my money, you change my …
14639	neutral	@AmericanAir we have 8 ppl so we need 2 know h…

[14640 rows x 2 columns]

3 TRAINING THE MODEL

[16]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
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


# Tokenize the data
tokenizer = Tokenizer(num_words=5000)
tokenizer.fit_on_texts(data['text'])

# Pad the sequences to the same length
X = pad_sequences(tokenizer.texts_to_sequences(data['text']), maxlen=256)

# Encode the target labels as integers (0 for Negative, 1 for Neutral, 2 for
↪Positive)
y = data['airline_sentiment'].map({'negative': 0, 'neutral': 1, 'positive': 2}).
↪values
```

[18]:
```
from keras.layers import Bidirectional, Embedding, LSTM, Dense
from keras.regularizers import l2

model = Sequential()
model.add(Embedding(input_dim=5000, output_dim=128))  # Remove
↪kernel_regularizer
model.add(Bidirectional(LSTM(128, return_sequences=True,
↪kernel_regularizer=l2(0.01))))  # Apply kernel_regularizer here
model.add(Bidirectional(LSTM(64, kernel_regularizer=l2(0.01))))  # Apply
↪kernel_regularizer here
model.add(Dense(3, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
↪metrics=['accuracy'])
model.fit(X_train, y_train, epochs=12, batch_size=64)
loss, accuracy = model.evaluate(X_test, y_test)
print('Test loss:', loss)
print('Test accuracy:', accuracy)
```

Conclusion:

Our sentiment analysis project in marketing empowers us to harness the power of natural language processing and data analytics to understand customer sentiments more profoundly. By classifying feedback into positive, neutral, and negative categories, we can unlock valuable insights that guide our marketing strategies and decision-making processes. These insights help us improve our products, enhance customer satisfaction, and fine-tune our marketing campaigns, ultimately driving business growth.