الجامعة الأمريكية في الإمارات

American University in the Emirates

Analysing Student Feedback Using Artificial Intelligence Techniques in Python

Mohammed Faaiz Jaffar

September 2023

ACKNOWLEDGEMENT

I express my sincere gratitude towards **Ms. Haneen M. Anbar, VP for Administrative Affairs at AUE,** for giving me the opportunity to showcase my skills and interest in AI with Python. This assignment was immensely interesting for me. I learnt a lot through this assignment about using python for analysing student feedback sentiments, subjects of comments and student opinions.

With Humble Regards,

Mohammed Faaiz Jaffar [faaizjaffar@outlook.com, +971545002737]

ABSTRACT

This report presents an in-depth exploration of the application of Artificial Intelligence (AI) techniques in Python for the analysis of student feedback. Authored by Mohammed Faaiz Jaffar, this endeavor demonstrates the capability to proficiently employ Python-based AI methodologies, with a specific focus on Natural Language Processing (NLP) tasks and the extraction of valuable insights from textual data.

The dataset under consideration comprises student feedback comments gathered from course evaluations. Each comment is intricately linked with pertinent information, encompassing the academic year, semester, program, and more.

DEPENDENCIES

- Pandas
- numpy
- Matplotlib
- Textblob
- Googletrans
- Traceback
- gensim
- ipython
- spellchecker

PRE-DESIGN

TASK DESCRIPTION

As part of the application process, candidates are required to demonstrate their proficiency in Artificial Intelligence techniques using Python by completing a practical task involving the analysis of student feedback. This task is designed to assess candidates' ability to work with text data, perform Natural Language Processing (NLP) tasks, and extract insights using Python.

DATASET CONTENTS

The dataset provided includes student feedback comments collected from course evaluations. Each feedback comment is associated with a course and includes information about the semester and the student's program. The dataset has been provided in XLSX (Excel) format and contains the below columns:

- AcademicYear
- SemesterName
- ParticipantID
- DegreeLevel
- Gender
- Nationality
- StudentProgram
- CourseCode
- CourseName
- QuestionText
- QuestionType
- ParticipantResponse

DESIGN

SENTIMENT ANALYSIS OF COMMENTS

LOADING DATA

The library ‘pandas’ is used to read the excel file.

```
df = pd.read_excel('C:/Users/faaiz/Downloads/Hello/AI_Engineer_Dataset_Task_1.xlsx')

user_comments_df = df[df['QuestionType'] == 'User Comment']
```

The user comments raw data is placed into ‘user_comments_df’.

SENTIMENT ANALYSIS MODEL

Sentiment Analysis is done through a custom function get_sentiment(text):

String texts (non-string values filtered out) are filtered and fed into the TextBlob() function. After textblob function executes it assigns several attributes to the analysis (analysis = TextBlob(text)), Under the sentiment attribute of the analysis, there is a polarity attribute that when checked returns a number between -1 and 1. Positive number = positive sentiments and vice versa.

```
def get_sentiment(text):
    if isinstance(text, str):
        analysis = TextBlob(text)
        if analysis.sentiment.polarity > 0:
            return 'Positive'
        elif analysis.sentiment.polarity < 0:
            return 'Negative'
        else:
            return 'Neutral'
    else:
        return 'NaN'
```

Now the get_sentiment() is applied to each comment. And the result in placed in the 'Sentiment' column of the user_comments_df matrix (to corresponding comment).

DATA PLOTTING AND VISUALIZATION

The positive, negative and neutral (and NaN) are counted and a bar graph is plotted with the counts on the Y axis and the number of comments in positive, neutral, and negative sentiments are plotted as bars on the x-axis. The Graph is labelled.

```
sentiment_counts = user_comments_df['Sentiment'].value_counts()
plt.bar(sentiment_counts.index, sentiment_counts.values)
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.title('Sentiment Distribution of User Comments')
plt.show()


summary_stats = user_comments_df['Sentiment'].value_counts()
print(summary_stats)
```

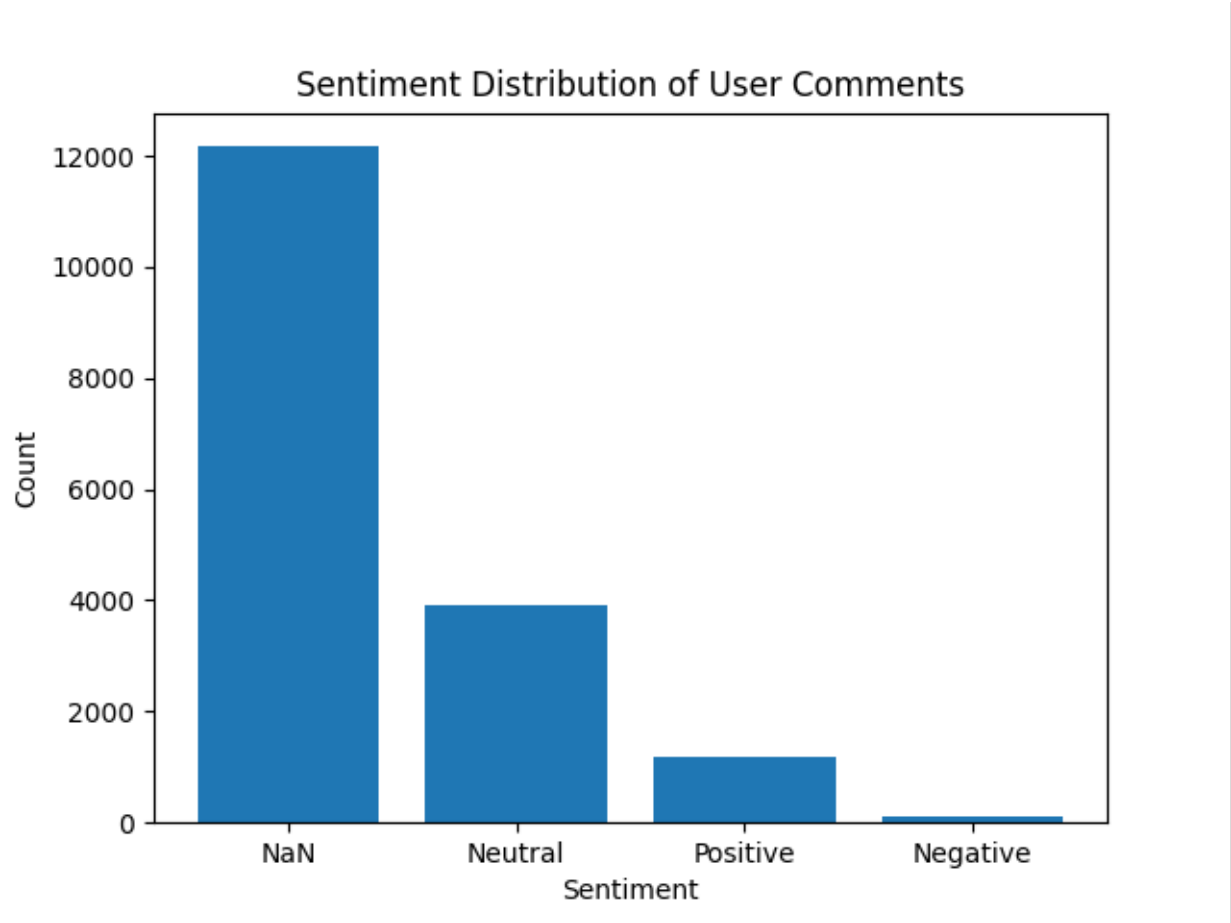


Figure 1 (Sentiment Analysis of Student Comments)

Note: The vast amount of NaN/Null values are due to a large proportion of the comments being in **Arabic**. To tackle this issue google's translate python library can be used. However the translation takes a large amount of time and hence has not been visualized due to time constraints. The code for translated dataframe is below. **A lemmatizer and tokenizer is used in the final code to increase accuracy.**

```
df = pd.read_excel('C:/Users/faaiz/Downloads/Hello/AI_Engineer_Dataset_Task_1.xlsx')

# bringing only comments out into the dataframe
user_comments_df = df[df['QuestionType'] == 'User Comment']

translator = Translator()

# custom function for sentiment analysis
def get_sentiment(text):
    if isinstance(text, str):  # check if text string.
        try:
            # Translate the text to English
            translated_text = translator.translate(text, src='ar', dest='en').text
            analysis = TextBlob(translated_text)  # change text to translated_text if uncommented translation code
            if analysis.sentiment.polarity > 0:
                return 'Positive'
            elif analysis.sentiment.polarity < 0:
                return 'Negative'
            else:
                return 'Neutral'
        except Exception as e:
            pass
    else:
        return 'NaN'  # arabic comments and Null values being returned as NaN

# applying sentiment analysis and placing it under sentiment column
user_comments_df['Sentiment'] = user_comments_df['ParticipantResponse'].apply(get_sentiment)

# data visualization
sentiment_counts = user_comments_df['Sentiment'].value_counts()
plt.bar(sentiment_counts.index, sentiment_counts.values)
plt.xlabel('Sentiment')
plt.ylabel('Count')
plt.title('Sentiment Distribution of User Comments')
plt.show()
```

AUTHORS INTERPRATION

I believe the algorithm will be even more accurate when translate is used along with removing stopwords from nltk and using SpellChecker. The large amount of positive sentiments is a great fact for the university itself. However, it is important to consider if the students were explicitly told that it is an anonymous survey. Many students tend to believe there is some type of tracking that might make them susceptible if they actually had negative remarks to put in. With translate used the sentiment analysis would have been very close to accurate. Also by looking at the large amount Null values, it is evident a large population of the students are native Arabic speakers. Most Arabic used here in not traditional, hence symbolizing the diversity within Arabic speakers themselves which is great for a lively atmosphere. **A lemmatizer is used in the final code to increase accuracy.**

LDA

DATA LOADING

Data loading is done in the same way as shown previously.

```
df = pd.read_excel('C:/Users/faaiz/Downloads/Hello/AI_Engineer_Dataset_Task_1.xlsx')

# bringing only comments out into the dataframe
user_comments_df = df[df['QuestionType'] == 'User Comment']
```

TOKENIZATION

```
# custom function includes translate, tokenizing (into single words) all the user comments
def translate_and_tokenize(text):
    if isinstance(text, str):  # Check if 'text' is a string.
        try:
            # uncomment below line of code to feed translation
            #translated_text = translator.translate(text, src='ar', dest='en').text
            # using word_tokenize function from nltk
            tokens = word_tokenize(text) # change text to translated_text if translation line is uncommented
            return [word.lower() for word in tokens]
        except Exception as e:
            pass
    else:
        return []

# applying tokenization and placing tokenized comments under tokenzied comments column
user_comments_df['TokenizedComments'] = user_comments_df['ParticipantResponse'].apply(translate_and_tokenize)
```

The user comments are broken down into words so that topics can be analysed later. (translation is ignored for now due to time constraint) tokenized comments are placed under tokenized comments column.

LDA

```
documents = user_comments_df['TokenizedComments'].tolist()

dictionary = corpora.Dictionary(documents)

# create a document-term matrix
doc_term_matrix = [dictionary.doc2bow(doc) for doc in documents]

#  Latent Dirichlet Allocation
lda_model = LdaModel(doc_term_matrix, num_topics=5, id2word=dictionary, passes=20)

# increase num_topics and passes for accuracy but increased computing time
topics = lda_model.print_topics(num_words=5)

# Visualize the topics using pyLDAvis
vis_data = gensimvis.prepare(lda_model, doc_term_matrix, dictionary)
pyLDAvis.display(vis_data)

# Print the top keywords for each topic
topics = lda_model.print_topics(num_words=10)  # adjust number of keywords as needed
```

‘documents’ is a list where each element is the tokenized comment of each student.

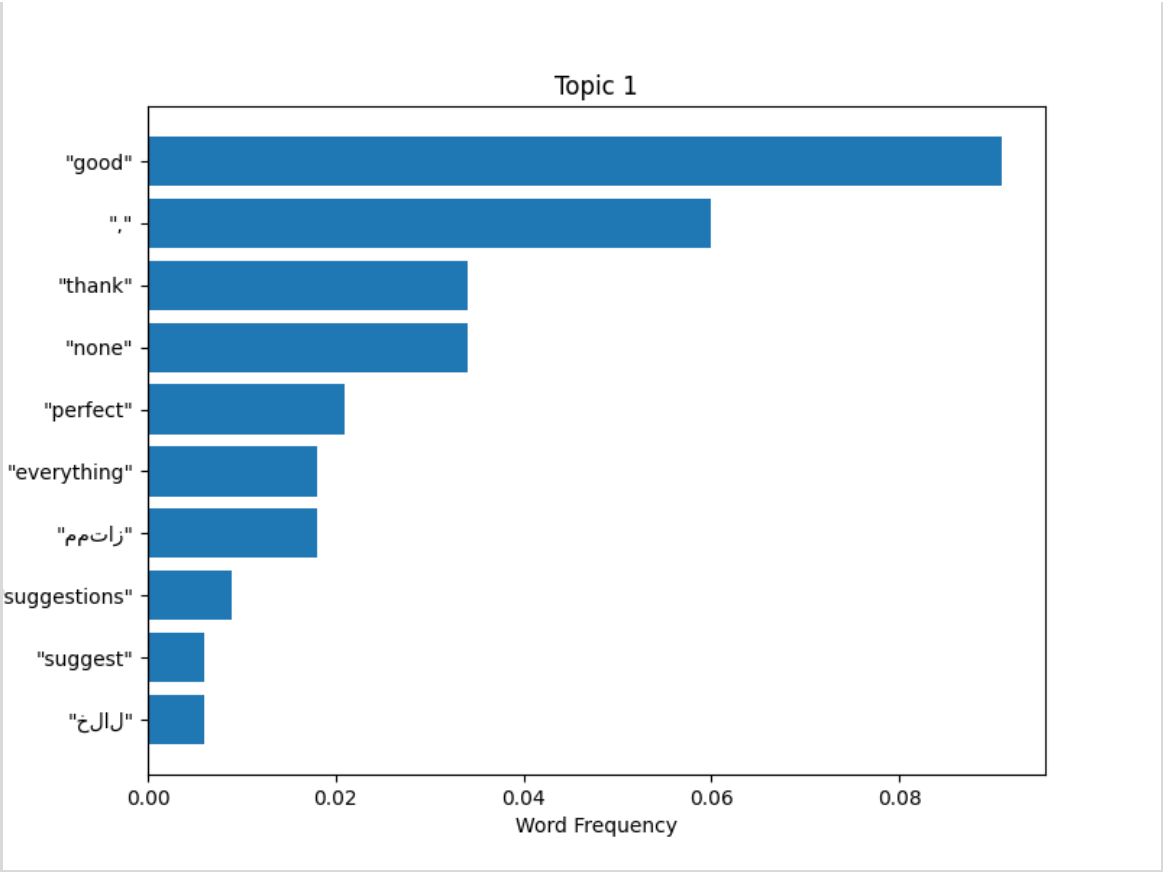
corpora from gensim consist of the dictionary function that maps each token (word) to a unique ID

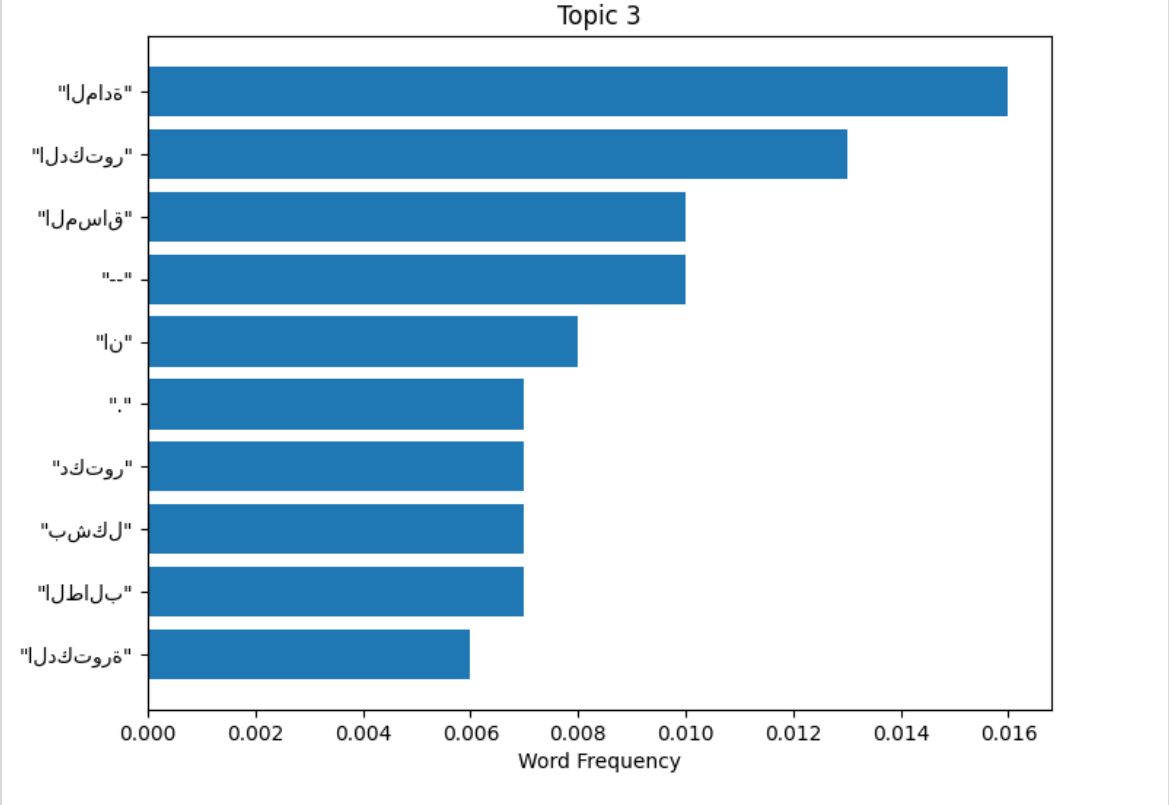
now a comment (document) and term (token/word) matrix is created where in each comment has specific unique ID (of term/word) and word frequency (in the same comment) [Bag of Words method].

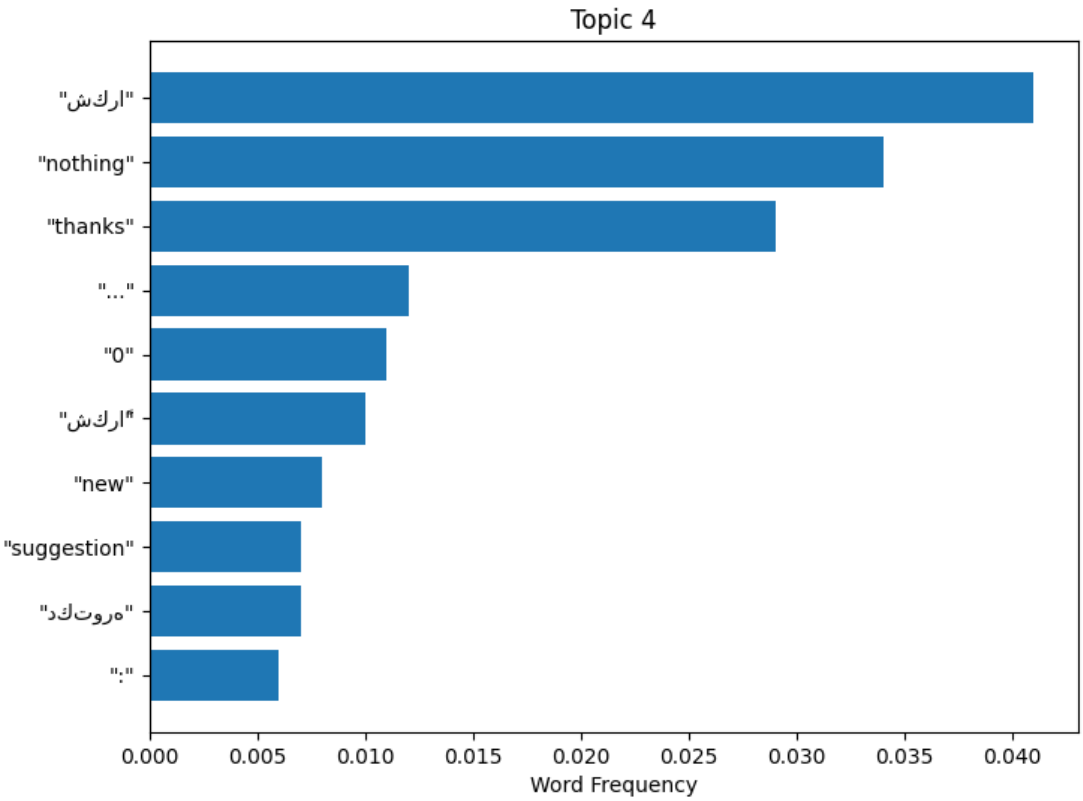
Later LDA is used on the document-term matrix. With number of topics 5 and 20 passes.

```
# Plot the top keywords for each topic


for i, topic in enumerate(topics):
    top_words = [val.split('*')[1].strip() for val in topic[1].split('+')]
    word_freq = [float(val.split('*')[0]) for val in topic[1].split('+')]

    plt.figure(figsize=(8, 6))
    plt.barh(top_words, word_freq)
    plt.gca().invert_yaxis()
    plt.xlabel('Word Frequency')
    plt.title(f'Topic {i + 1}')
    plt.show()
```

A total of 5 plots are made.


Topic 1
"good"
","
"thank"
"none"
"perfect"
"everything"
"ممتاز"
suggestions"
"suggest"
"خلال"
0.00
0.02
0.04
0.06
0.08
Word Frequency


Topic 3
"المادة"
"الدكتور"
"المساق"
"--"
"ان"
"."
"دكتور"
"بشكل"
"الطالب"
"الدكتورة"
0.000
0.002
0.004
0.006
0.008
0.010
0.012
0.014
0.016
Word Frequency



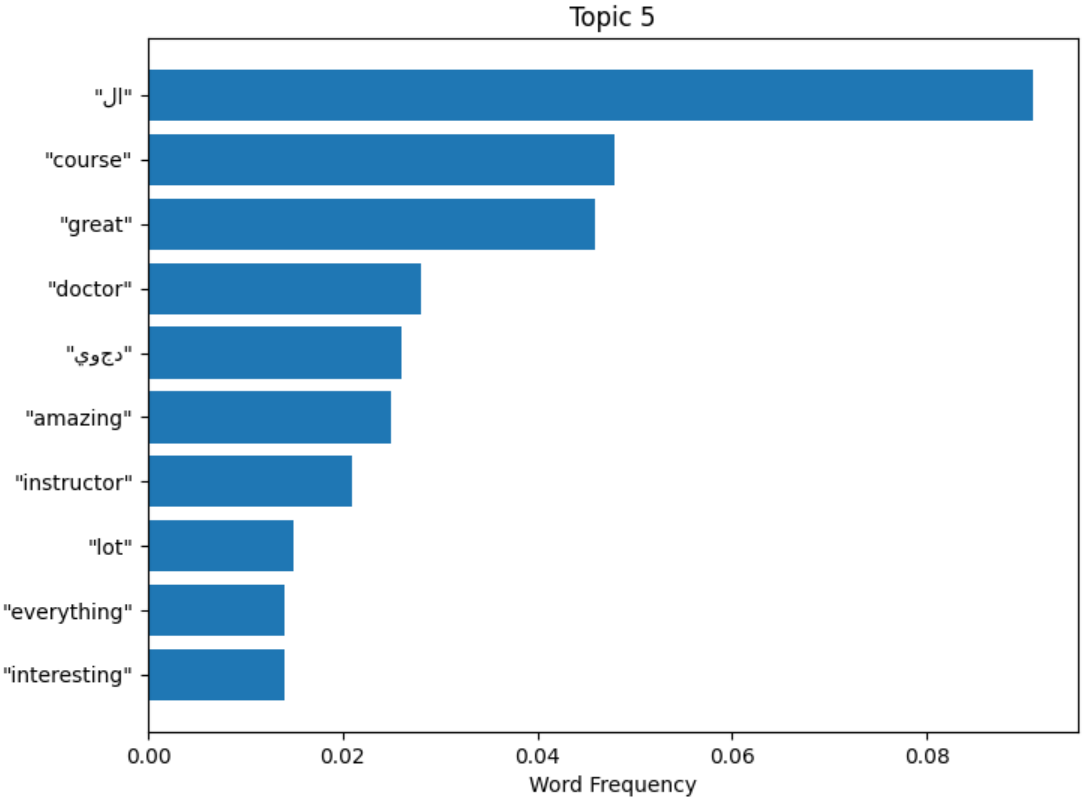


Figure 2-6 Topic Frequency using LDA

AUTHOR'S INTERPRETATION

From the words I can understand a positive outcome of the courses. Again the use of spellcheckers and translate can increase the accuracy of interpretation by a lot. However, it has been avoided for now due to time constraints. With the use of instructor and doctor, dr. it is obvious that quite a lot of the students have to talk about their instructors. whenever doctor/dr. is used it refers that many students are referring to a singular professor respectively. Also the lack of translation brings in inaccuracy in the system. **A lemmatizer is used in the final code to increase accuracy.**

OPINION MINING

DATA LOADING

Here the Ratings and Comments will be both used together for forming an opinion analysis.

```
df = pd.read_excel('C:/Users/faaiz/Downloads/Hello/AI_Engineer_Dataset_Task_1.xlsx')


relevant_question_types = ['Rating', 'User Comment']
filtered_df = df[df['QuestionType'].isin(relevant_question_types)]
```

The dataset will consist of Rating wherever relevant and comment wherever its relevant.

SENTIMENT ANALYSIS AND OPINION MINING

```
def analyze_sentiment(text):
    try:
        if isinstance(text, float):  # Check if the value is a float (e.g., NaN)
            return 'Neutral'
        #sentiment analysis using TextBlob library
        analysis = TextBlob(str(text))

        if analysis.sentiment.polarity > 0:
            return 'Positive'
        elif analysis.sentiment.polarity < 0:
            return 'Negative'
        else:
            return 'Neutral'
    except:
        return 'Error'  # Handle any exceptions during analysis
```

Sentiment analysis done as shown in the first part of the assignment.

```
# applying sentiment analysis to each element in participant response column and adding
# sentiment under sentiment column
filtered_df['Sentiment'] = filtered_df['ParticipantResponse'].apply(analyze_sentiment)

print("Sentiment Analysis Results:")
print(filtered_df[['ParticipantResponse', 'Sentiment']])

# Form analysis by counting sentiments
sentiment_counts = filtered_df['Sentiment'].value_counts()
print("\nSentiment Statistics:")
print(sentiment_counts)
```

DATA VISUALIZATION

```
Sentiment Analysis Results:
       ParticipantResponse Sentiment
0                 Disagree   Neutral
1        Strongly Disagree  Positive
2        Strongly Disagree  Positive
3        Strongly Disagree  Positive
4        Strongly Disagree  Positive
...                    ...       ...
180964                 NaN   Neutral
180965               Agree   Neutral
180966      Strongly Agree  Positive
180967      Strongly Agree  Positive
180968      Strongly Agree  Positive

[180969 rows x 2 columns]

Sentiment Statistics:
Sentiment
Positive    107117
Neutral      73744
Negative       108
```

AUTHOR'S INTERPRETATION

The sentiment statistics show a fair amount of information taken from the ratings and comments in the student feedback. Positive and Neutral sentiments are proportionally higher than negative. However, I doubt the accuracy as some questions could be rhetorical or negative to which the answers would be Strongly Agree/Agree/Neutral causing inaccuracy. In order the counter this the questions have to undergo sentiment analysis first to see which questions are positive and which are negative. Positive ratings to positive questions should increase true positive sentiment while positive ratings to negative questions should increase true negative sentiment. These changes could be made however it would take more computation and due to time constraint has been avoided for now. **A lemmatizer is used in the final code to increase accuracy.**

APPENDICES

https://github.com/fjotb/AUE