Chapter 5

Introduction to Sentiment Analysis

What is sentiment analysis?

- Sentiment analysis, also called **opinion mining**, is the process of understanding the opinion of an author about a subject.

What goes into a sentiment analysis system?

- **3 elements** in a sentiment analysis system that are usually depending on the context:
  - **First**: Opinion / emotion
    - An opinion (polarity) can be positive, neutral, or negative.

- An emotion could be qualitative (like joy, surprise, or anger) or quantitative (like rating a movie on the scale from 1 to 10).

What goes into a sentiment analysis system?

- **Second**: Subject
  - What is the **subject** that is being talked about?
    - Such as a book, a movie, or a product.
    - For example: "The camera on this phone is great but its battery life is rather disappointing."
- **Third**: Opinion holder (the person or organization claiming an opinion)
  - By whom?

Why sentiment analysis?

- Sentiment analysis has many **practical applications** such as:
  - **Social media monitoring**:
    - We don't just want to know if people are **talking about a brand**; we want to know how they are talking about it.
    - Social media isn't only source of information; we can also find sentiment on blogs, and the news.
  - **Brand monitoring**:
    - Most brands analyze all of these sources **to enrich their understanding** of
      - how customers **interact** with their brand,
      - what they are **happy** or **unhappy** about.
  - Sentiment analysis is **thus** very important in **brand monitoring**, and in fields such as customer and **product analytics** and **market research and analysis**.

Let's look at **movie reviews**!

https://www.imdb.com/

- The dataset: a sample of IMDB movie reviews that has two columns including one for the text of the review, and a second one called "label", which expresses the overall sentiment:
  - the category or class 1 means positive and 0 means negative.

data.head()

Text (review)	label
This short spoof can be found on Elite's Mille...	0
A singularly unfunny musical comedy that artif...	0
An excellent series, masterfully acted and dir...	1
The master of movie spectacle Cecil B. De Mill...	1
I was gifted with this movie as it had such a ...	0

How many positive and negative reviews?

- Call the .value_counts() method on the "label" column:
  - data.label.value_counts()
    - 0: the number of negative reviews
    - 1: the number of positive reviews

```
Number of positive and negative reviews:
0    20019
1    19981
Name: label, dtype: int64
```

Percentage of positive and negative reviews

- To see the number of positives and negatives as a percentage
  - Divide the expression by the number of rows that obtain with the **len()** method.
    - data.label.value.count() / len(data)

```
Proportion of positive and negative reviews:
0    0.500475
1    0.499525
Name: label, dtype: float64
```

How long is the longest review?

- Select the review column of the dataset, followed by **.str.len()**.
  - **str** is short for string.
  - Call the string function to transform the Series of reviews to a string.

```
length_reviews = movies.text.str.len()
type(length_reviews)
```

```
<class 'pandas.core.series.Series'>
```

  - Find the text review with max length

```
max(length_reviews)
```

การหาความยาวของรีวิวมีความสำคัญอย่างไร?

Let's practice!

How many positive and negative reviews are there?

- As a first step in a sentiment analysis task, similar to other data science problems,
  - to explore the dataset in more detail.

- Frist, load dataset

```
import pandas as pd
movies = pd.read_csv('.\ch5\\train.csv')
```

How many positive and negative reviews are there?

- Find the number of positive and negative reviews in the movies dataset.

```
# Find the number of positive and negative reviews
print('Number of positive and negative reviews: ',
movies.label.value_counts())
```

- Find the percentage of positive and negative reviews in the dataset.

```
# Find the proportion of positive and negative reviews
print('Proportion of positive and negative reviews: ',
movies.label.value_counts() / len(movies))
```

Longest and shortest reviews

- Now your task is to explore the review column in more detail.
- Use the **text** column of the **movies** dataset to find the length of the longest review.

```
length_reviews = movies.text.str.len()

# How long is the longest review
print(max(length_reviews))
```

Sentiment analysis types and approaches

Levels of granularity

- **Document** level
  - Look at the **whole review** of a product
- **Sentence** level
  - Refers to determining whether **the opinion expressed in each sentence** is positive, negative, or neutral
- **Aspect** level
  - refers to expressing opinions about **different features of a product**.
  - Imagine a sentence: "The camera in this phone is pretty good but the battery life is disappointing."
    - It expresses both positive and negative opinions about a phone and want to be able to say which features of the product clients like and which they don't.

Type of sentiment analysis algorithms

- Rule or lexicon based
  - This methods most commonly have a predefined list of words with a valence score. For example, *nice could be +2, good +1, terrible -3*, and so on.
  - The algorithm matches the words from the lexicon to the words in the text and either **sums** or **averages** the **scores**. For example, 'Today was a good day.'

    Today: 0, was: 0, a: 0, good: +1, day: 0
    Total valence: +1

  - Each word gets a score, and to get the total valence. So, we have a positive sentence.
- Automated systems, which are based on machine learning
  - Usually modeled as a classification problem where using some historical data with known sentiment,
  - Predict the sentiment of a new piece of text.

What is the valence of a sentence?

- Calculate the valence score of a text, using Python's **TextBlob** library.
- A TextBlob object is like a Python string, which has obtained some natural language processing skills.

```
text = 'Today was a good day'
my_valence = TextBlob(text)
my_valence.sentiment
```

```
Sentiment(polarity=0.7, subjectivity=0.6000000000000001)
```

- The sentiment property returns a tuple:
  - First element is **polarity (ขั้วความรู้สึก)**,
    - measured from [-1.0 to 1.0], where -1.0 is very negative, 0 is neutral and +1.0 is very positive.
    - Our example 'Today was a good day' carries positive emotion and thus will have a positive polarity score: 0.7.
  - The second element is **subjectivity**,
    - measured from [0.0 to 1.0] where 0.0 is very objective (ความเห็นที่อยู่บนข้อเท็จจริงที่ทุกคนเห็นเหมือนกัน) and 1.0 is very subjective (ความเห็นในมุมของตนเองเป็นหลัก). So this example is subjective.

Automated or rule-based?

Automated/Machine Learning

- relies on having labeled historical data
- might take a while to **train**
- can be quite powerful

Rule/Lexicon-based

- rely on having manually **created rules** or dictionaries
- Different words might have different polarity in different contexts
- Can be quite fast

Let's practice I

Detecting the sentiment

- To detect the sentiment, including polarity and subjectivity of a given string using such a **rule-based approach** and **the TextBlob library** in Python.
- Import the required packages

```
# Import the required packages
from textblob import TextBlob
```

- Create a text blob object from the text string.

```
text = "You are so beautiful"
# Create a textblob object
blob_two_cities = TextBlob(text)
```

- Print out the polarity and subjectivity.

```
# Print out the sentiment
print(blob_two_cities.sentiment)
```

```
Sentiment(polarity=0.85, subjectivity=1.0)
```

What is the sentiment of a movie review?

- Import the required functionality.

```
# Import the required packages
from textblob import TextBlob
```

- Read/Load the review of the titanic movie (titanic.txt)

```
#Read TXT file
f = open(".\ch5\\titanic.txt", "r")
titanic = f.read()
```

- Create a text blob object from the titanic string.

```
# Create a textblob object
blob_titanic = TextBlob(titanic)
```

- Print out the result of its sentiment property.

```
# Print out its sentiment
print(blob_titanic.sentiment)
```

```
Sentiment(polarity=0.2024748060772906, subjectivity=0.4518248900857597)
```

Let's practice II

Import modules for sentiment analysis

- The re module provides operations for **regular expression matching**, useful for pattern and string search.
- pandas is one of the most widely used open-source tools for **data manipulation and analysis**.
- matplotlib is an easy-to-use, popular and comprehensive library in Python for creating visualizations.
- nltk is a comprehensive open-source platform for building applications to process human language data. It comes with powerful text processing libraries for typical Natural Language Processing (NLP) tasks like **cleaning, parsing, stemming, tagging, tokenization, classification, semantic reasoning**, etc.
- The VADER lexicon with **NLTK's SentimentIntensityAnalyzer** class is used to assign a **sentiment score** to each comment in the demo dataset.
  - Valence Aware Dictionary and Sentiment Reasoner (**VADER**) is a **lexicon** and **rule-based sentiment analysis toolset** with a focus on sentiments contained in general text applications like online comments, social media posts, and survey responses.

Import modules for sentiment analysis

- Load the necessary modules
- Download the ‘vader_lexicon’ for sentiment analysis
- Creates pd for referencing pandas
- Create plt for referencing matplotlib modules

```
import re
import pandas as pd
import matplotlib.pyplot as plt
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
```

Import demo data file and pre-process text

- Uses the read_excel method from pandas to load the demo input datafile into a panda dataframe.
- Add a new field row_id to this dataframe by incrementing the in-built index field. This row_id field serves as the unique key for this dataset to uniquely identify a row and will be used later in the code for merging two dataframes.

```
#df = pd.read_excel (r'Path where the Excel file is stored\File name.xlsx')
df = pd.read_excel (r".\\ch5\\TeamHealthRawDataForDemo.xlsx")
# adding an row_id field to the dataframe, which will be useful for joining later
df["row_id"] = df.index + 1
#print first 10 rows
print (df.head(10))
```

	Period	Manager	Team	Response	row_id
0	2019-Q1	Mgr 1	Team 1	We're a fun team that works well together and ...	1
1	2019-Q1	Mgr 1	Team 1	we have a sound and collaborative team focused...	2
2	2019-Q1	Mgr 1	Team 1	we work well as a team, we have fun together, ...	3
3	2019-Q1	Mgr 1	Team 1	I fell pretty good about the health of our tea...	4
4	2019-Q1	Mgr 1	Team 1	happy with team's overall health and good dyna...	5
5	2019-Q1	Mgr 1	Team 1	Solid	6
6	2019-Q1	Mgr 1	Team 1	The Team 2 team is a collaborative group prod...	7
7	2019-Q1	Mgr 1	Team 1	We have great teamwork. We have a lot of fun....	8
8	2019-Q1	Mgr 1	Team 1	We feel good about our teamwork, process, tech...	9
9	2019-Q1	Mgr 1	Team 2	A blast! Always working towards delivering mo...	10

Import demo data file and pre-process text

- Subset row_id and Response fields into a new dataframe, which is the input format required for by the SentimentIntensityAnalyzer class.
- **Cleans up the text data** by removing all non-alphabet characters and converting all text to lower case.

```
#create a new data frame with "id" and "comment" fields
df_subset = df[['row_id', 'Response']].copy()
#data clean-up
#remove all non-aphabet characters
df_subset['Response'] = df_subset['Response'].str.replace("[^a-zA-Z#]", " ")
#covert to lower-case
df_subset['Response'] = df_subset['Response'].str.casefold()
print (df_subset.head(10))
```

```
   row_id                                           Response
0       1  we re a fun team that works well together and ...
1       2  we have a sound and collaborative team focused...
2       3  we work well as a team  we have fun together  ...
3       4  i fell pretty good about the health of our tea...
4       5  happy with team s overall health and good dyna...
5       6                                              solid
6       7  the team   team is a  collaborative group prod...
7       8  we have great teamwork   we have a lot of fun ...
8       9  we feel good about our teamwork  process  tech...
9      10  a blast   always working towards delivering mo...
```

Generate **sentiment polarity scores**

- The sentiment lexicon in VADER is a list of lexical features like words and phrases labeled as positive or negative according to their semantic orientation.
- Its rule-based approach is especially good at detecting sentiments in common applications like social media posts, product or service reviews, and survey responses.
- Its **generates a numeric score** in the range of negative one (-1) to positive one (+1) to indicate the intensity of how negative or positive the sentiment is.
- This is called the **polarity score** and is implemented by the polarity_score method of the SentimentIntensityAnalyzer class.
  - The range of -1 to -0.5 indicates **negative** sentiment
  - The score greater than -0.5 and less than +0.5 indicates **neutral** sentiment
  - The range of +0.5 to 1 indicates **positive** sentiment

Generate sentiment polarity scores

- Create a dataframe for staging the output of the SentimentIntensityAnalyzer.polarity_scores method.

```
# set up empty dataframe for staging output
df1=pd.DataFrame()
df1['row_id']=['99999999999']
df1['sentiment_type']='NA999NA'
df1['sentiment_score']=0
```

Generate sentiment polarity scores

- Involve instantiating an object of the class SentimentIntensityAnalyzer and running a for-loop to iterate the polarity_scores method over each row of input text dataframe df_subset.
- Another for loop is embedded with the earlier loop to write the sentiment polarity score for each sentiment type to an intermediate dataframe.
- The three sentiment type values are:
  - *neg* for negative sentiment
  - *neu* for neutral sentiment
  - *pos* for positive sentiment
  - *compound* for an **overall score** that combines negative, positive, and neutral sentiments into a single score.

Generate sentiment polarity scores

- At the end of the for loop, clean the output dataframe by:
  - Deleting the dummy row from the output dataframe
  - Removing any duplicate rows that could potentially creep into the output dataframe
  - Filtering the output dataframe to only keep rows for sentiment type of compound

```
print('Processing sentiment analysis...')
sid = SentimentIntensityAnalyzer()
t_df = df1
for index, row in df_subset.iterrows():
    scores = sid.polarity_scores(row[1])
    for key, value in scores.items():
        temp = [key,value,row[0]]
        df1['row_id']=row[0]
        df1['sentiment_type']=key
        df1['sentiment_score']=value
        t_df=pd.concat([t_df,df1])
#remove dummy row with row_id = 99999999999
t_df_cleaned = t_df[t_df.row_id != '99999999999']
#remove duplicates if any exist
t_df_cleaned = t_df_cleaned.drop_duplicates()
# only keep rows where sentiment_type = compound
t_df_cleaned = t_df[t_df.sentiment_type == 'compound']
print(t_df_cleaned.head(10))
```

	row_id	sentiment_type	sentiment_score
0	1	compound	0.6597
1	2	compound	0.9287
2	3	compound	0.8122
3	4	compound	0.8225
4	5	compound	0.8271
5	6	compound	0.1531
6	7	compound	0.9382
7	8	compound	0.9381
8	9	compound	0.9468
9	10	compound	0.5519

Generate sentiment polarity scores

- Merge the output dataframe t_df_cleaned with the input dataframe df using the field row_id.
- This dataframe merge operation in Python is conceptually similar to performing a join on two database tables in SQL.
- The merged dataframe will have the following fields:
  - Period
  - Manager
  - Team
  - Response
  - Row_id
  - Sentiment_type
  - Sentiment_score

```python
#merge dataframes
df_output = pd.merge(df, t_df_cleaned, on='row_id', how='inner')
print(df_output.head(10))
```

```
    Period Manager    Team                                           Response  \
0  2019-Q1   Mgr 1  Team 1  We're a fun team that works well together and ...
1  2019-Q1   Mgr 1  Team 1  we have a sound and collaborative team focused...
2  2019-Q1   Mgr 1  Team 1  we work well as a team, we have fun together, ...
3  2019-Q1   Mgr 1  Team 1  I fell pretty good about the health of our tea...
4  2019-Q1   Mgr 1  Team 1  happy with team's overall health and good dyna...
5  2019-Q1   Mgr 1  Team 1                                             Solid
6  2019-Q1   Mgr 1  Team 1  The Team 2 team is a  collaborative group prod...
7  2019-Q1   Mgr 1  Team 1  We have great teamwork.  We have a lot of fun....
8  2019-Q1   Mgr 1  Team 1  We feel good about our teamwork, process, tech...
9  2019-Q1   Mgr 1  Team 2  A blast!  Always working towards delivering mo...

   row_id sentiment_type  sentiment_score
0       1       compound           0.6597
1       2       compound           0.9287
2       3       compound           0.8122
3       4       compound           0.8225
4       5       compound           0.8271
5       6       compound           0.1531
6       7       compound           0.9382
7       8       compound           0.9381
8       9       compound           0.9468
9      10       compound           0.5519
```

Questions

Reference:

https://campus.datacamp.com/

https://www.red-gate.com/simple-talk/development/data-science-development/sentiment-analysis-python/?fbclid=IwAR2saZfrAYF3CGuPiMgBongQPuRyrv3olMa7rCR1CDhPQ1Q3kpoYBjCDrME