Sentiment Analysis

Sentiment Analysis is a use case of **Natural Language Processing (NLP)** and comes under the category of **text classification**. To put it simply, Sentiment Analysis involves classifying a text into various sentiments, such as positive or negative, Happy, Sad or Neutral, etc. Thus, the ultimate goal of sentiment analysis is to decipher the underlying mood, emotion, or sentiment of a text. This is also known as **Opinion Mining**.

Sentiment analysis is the process of analyzing digital text to determine if the emotional tone of the message is positive, negative, or neutral. Today, companies have large volumes of text data like emails, customer support chat transcripts, social media comments, and reviews. Sentiment analysis tools can scan this text to automatically determine the author's attitude towards a topic. Companies use the insights from sentiment analysis to improve customer service and increase brand reputation.

Types of Sentiment Analysis

Sentiment analysis focuses on the polarity of a text (*positive, negative, neutral*) but it also goes beyond polarity to detect specific feelings and emotions (*angry, happy, sad*, etc), urgency (*urgent, not urgent*) and even intentions (*interested v. not interested*).

- **Fine-grained Sentiment Analysis**: This goes beyond just positive, negative, or neutral. It involves very specific ratings, like a 5-star rating, for example.
- **Emotion detection**: This aims to detect emotions like happiness, frustration, anger, sadness, etc. The biggest challenge here is being able to accurately identify these emotions in text.

- **Aspect-based Sentiment Analysis**: This is generally used to understand specific aspects of a certain product or service. For example, in a review like "The battery life of this phone is great, but the screen is not very clear", the sentiment towards the battery life is positive, but it's negative towards the screen.
- **Multilingual sentiment analysis**: This can be particularly challenging because the same word can convey different sentiments in different languages.
- **Intent Analysis**: This goes a step further to understand the user's intention behind a certain statement. For example, a statement like "I would need a car" might indicate a purchasing intent.

Sentiment Analysis in Python using Text Blob

Text Blob is a Python library for Natural Language Processing. Using Text Blob for sentiment analysis is quite simple. It takes text as an input and can return **polarity** and **subjectivity** as outputs.

Polarity determines the sentiment of the text. Its values lie in [-1,1] where -1 denotes a highly negative sentiment and 1 denotes a highly positive sentiment.

Subjectivity determines whether a text input is factual information or a personal opinion. Its value lies between [0,1] where a value closer to 0 denotes a piece of factual information and a value closer to 1 denotes a personal opinion.

CREATIVE INSTITUTE DATA SCIENCE

Read the necessory library

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
comments = pd.read_csv('UScomments.csv',on_bad_lines='skip')
```

```
comments.shape
```

```
comments.head()
```

```
comments.isnull().sum()
```

```
comments.dropna(inplace=True)
```

```
comments.isnull().sum()
```

sentiment analysis

```
from textblob import TextBlob
```

```
comments.head()
```

```
TextBlob("Logan Paul it's yo big day !!!!!").sentiment.polarity
```

```
comments.shape
```

```
sample_df = comments[0:1000]
```

```
sample_df.shape
```

```
polarity=[]

for i in comments['comment_text']:
    try:
        polarity.append(TextBlob(i).sentiment.polarity)

    except:
        polarity.append(0)
```

```
len(polarity)
```

```
comments['polarity']  = polarity
```

```
comments.head(5)
```

```
filter1 = comments['polarity']==1
```

```
comments_positive = comments[filter1]
```

```
filter2 = comments['polarity']==-1
```

```
comments_negative = comments[filter2]
```

```
comments_positive.head(5)
```

WORDCLOUD analysis

A word cloud (also known as a tag cloud or text cloud) is a visual representation of a text, in which the words appear bigger the more often they are mentioned.

Word clouds are great for visualizing unstructured text data and getting insights on trends and patterns.


```
from wordcloud import WordCloud,STOPWORDS
```

```
set(STOPWORDS)
```

need to generate positive wordcloud

```
comments['comment_text']
```

check the data type of text

```
type(comments['comment_text'])
```

we have seen that there is pandas series

need to convert in string format

now find total positive comment

```
total_comments_positive=" ".join(comments_positive['comment_text'])
```

```
type(total_comments_positive)
```

now generate cloud of positive comment and remove stopwords

```
wordcloud=WordCloud(stopwords=set(STOPWORDS)).generate(total_comments_positive)
```

now visualize the wordcloud

```
plt.imshow(wordcloud)
plt.axis('off')
```

now also find negative comment and create cloud

```
total_comments_negative = ' '.join(comments_negative['comment_text'])
```

```
wordcloud2 =
WordCloud(stopwords=set(STOPWORDS)).generate(total_comments_negative)
```

```
plt.imshow(wordcloud2)
plt.axis('off')
```

Perform Emoji's Analysis

For analysis emoji install emoji python package

```
pip install emoji==2.2.0
```

```
import emoji
```

```
emoji.__version__
```

```
comments['comment_text'].head(6)
```

```
comment='trending ☺'
```

now extract emoji from this data

```
[i for i in comment if i in emoji.EMOJI_DATA]
```

```
all_emoji_list=[]

for i in comments['comment_text'].dropna():

    for j in i:
        if j in emoji.EMOJI_DATA:
            all_emoji_list.append(j)
```

```
all_emoji_list[0:10]
```

now count most common emoji used

import collection counter method

```
from collections import Counter
```

```
Counter(all_emoji_list).most_common(10)
```

Now use 0 postion value

```
Counter(all_emoji_list).most_common(10)[0]
```

We have tupple data and we know that how to extract data from tupple

```
Counter(all_emoji_list).most_common(10)[0][0]
```

```
Counter(all_emoji_list).most_common(10)[1][0]
```

```
Counter(all_emoji_list).most_common(10)[2][0]
```

```
emojis=[Counter(all_emoji_list).most_common(10)[i][0] for i in range(10)]
```

```
emojis
```

now count the emoji

```
Counter(all_emoji_list).most_common(10)[0][1]
```

```
Counter(all_emoji_list).most_common(10)[1][1]
```

```
freqs=[Counter(all_emoji_list).most_common(10)[i][1] for i in range(10)]
```

```
freqs
```

```
sns.barplot(x=emojis,y=freqs)
```

Also we can see with plotly also

```
import plotly.graph_objs as go
from plotly.offline import iplot
```

```
trace = go.Bar(x=emojis , y=freqs)
```

```
iplot([trace])
```