NLP Assignment 3

Sentiment Analysis using Textblob or Vader

Meghananjali Remala

Z23701551

Task: Using Textblob or Vader (or both) for sentiment analysis.

There are 1000 positive and 1000 negative processed sentiment analysis text pieces in the dataset (unzip using 7-zip)

sentiment analysis texts.7z Download sentiment analysis texts.7z

Please follow the demo of Textblob or Vader to conduct sentiment analysis on the dataset (pick any one of the tools). You can also use both tools and compare their performance (optional).

Submit a report including the screen shots of running code and the generated results.

Steps:

1: Install Anaconda

2: Launch Jupyter Notebook

3: Verify Python and Pip Versions (Anaconda Prompt)

- Open the Anaconda Prompt, which is a command-line tool provided by Anaconda.
- To check the version of Python installed, run the command: python --version
- To check the version of Pip (Python package manager) installed, run the command: pip –version

4: Create a New Jupyter Notebook

- Within Jupyter Notebook, create a new notebook by selecting "New" and then "Python 3"

5: Install Python Packages (In Jupyter Notebook)

- In your Jupyter Notebook, you can easily install Python packages using the built-in magic command !. For example, to install the "textblob" library, you can run:
  !pip install textblob
- Similarly, to install the "vaderSentiment" library, you can run:
  !pip install vaderSentiment

Using TextBlob:

Code:

```
#Using TextBlob

from textblob import TextBlob

import os
```

```
# Paths to your positive and negative text files

positive_directory = "C://Users//dell//Desktop//Univ docs//Fau Univ docs//2nd Sem
FAU//NLP//txt_sentoken//pos"

negative_directory = "C://Users//dell//Desktop//Univ docs//Fau Univ docs//2nd Sem
FAU//NLP//txt_sentoken//neg"

# To keep track of sentiment scores

pos_sentiment_scores = []

neg_sentiment_scores = []

# Function to analyze sentiment in a text file using TextBlob

def analyze_textblob_sentiment(text):

    analysis = TextBlob(text)

    return analysis.sentiment.polarity

# Loop through positive text files

for filename in os.listdir(positive_directory):

    if filename.endswith(".txt"):

        with open(os.path.join(positive_directory, filename), "r") as file:

            text = file.read()

            sentiment_score = analyze_textblob_sentiment(text)

            pos_sentiment_scores.append(sentiment_score)

# Loop through negative text files

for filename in os.listdir(negative_directory):

    if filename.endswith(".txt"):

        with open(os.path.join(negative_directory, filename), "r") as file:

            text = file.read()

            sentiment_score = analyze_textblob_sentiment(text)

            neg_sentiment_scores.append(sentiment_score)

avg_pos_sentiment = sum(pos_sentiment_scores) / len(pos_sentiment_scores)

avg_neg_sentiment = sum(neg_sentiment_scores) / len(neg_sentiment_scores)


print("Sentiment scores of 1000 positive files",pos_sentiment_scores)

print("\nSentiment scores of 1000 negative files",neg_sentiment_scores)
print("Number of Positive files:",len(pos_sentiment_scores))

print("Number of Negative files:",len(neg_sentiment_scores))

print(f"Average Positive Sentiment: {avg_pos_sentiment}")

print(f"Average Negative Sentiment: {avg_neg_sentiment}")
```

Code Explanation:

Data Source:
- The text data is divided into two categories: positive and negative text files.
- Positive and negative text files are stored in separate directories.

Sentiment Analysis Function:
- A function, **analyze_textblob_sentiment**, is defined to evaluate the sentiment of a given text using TextBlob.
- TextBlob is a library that can assess the sentiment of text, indicating whether it's more positive or negative.

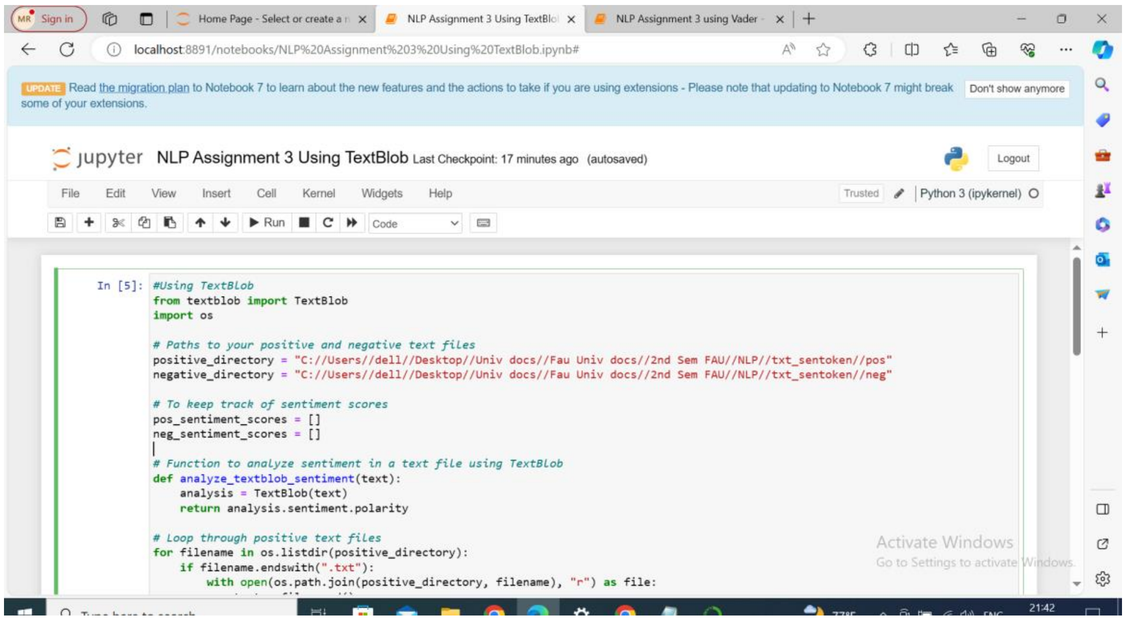
Processing Positive Texts:
- The code processes each text file in the "positive" category:
- It reads the text from the file.
- It uses TextBlob to analyze the sentiment of the text.
- The sentiment score, ranging from very negative to very positive, is recorded.
- These scores are saved in the **pos_sentiment_scores** list.

Processing Negative Texts:
- A similar process is applied to the "negative" text files:
- The text is read from each file.
- Sentiment analysis is performed using TextBlob.
- The scores are recorded in the **neg_sentiment_scores** list.

Calculating Averages:
- The code calculates the average sentiment scores for both positive and negative text files.
- This helps to understand the overall sentiment for each category.

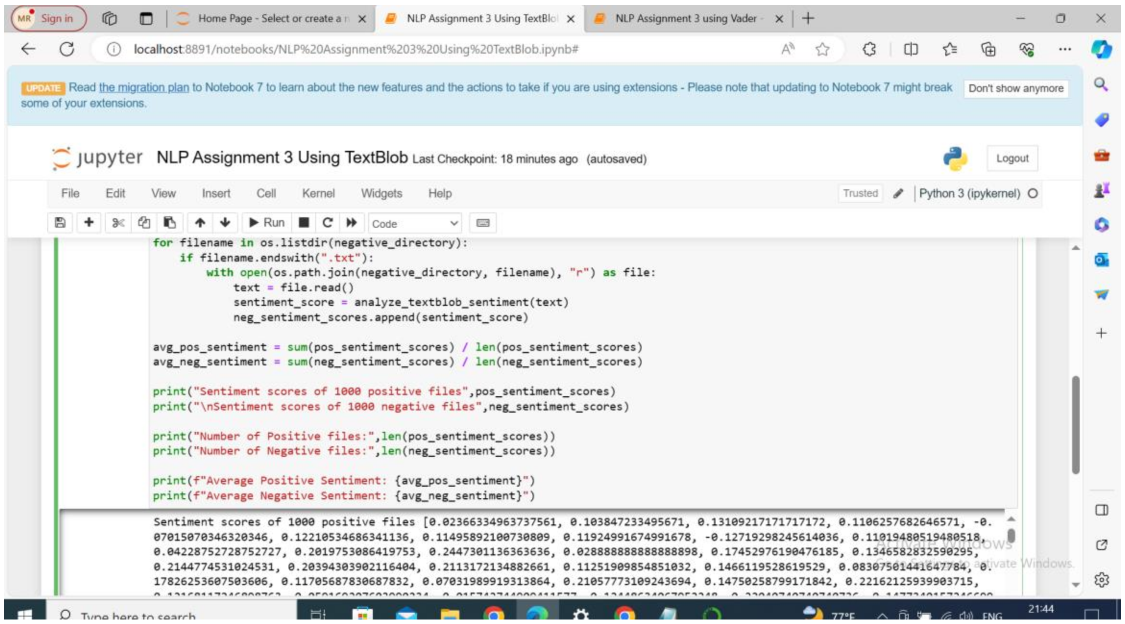
Displaying Results:
- The code presents the sentiment scores for individual files in both categories.
- It also shows the number of files in each category (positive and negative).
- Finally, it displays the average positive and negative sentiment scores.

Output Snapshots:



NLP Assignment 3 Using TextBlob Last Checkpoint: 17 minutes ago (autosaved)

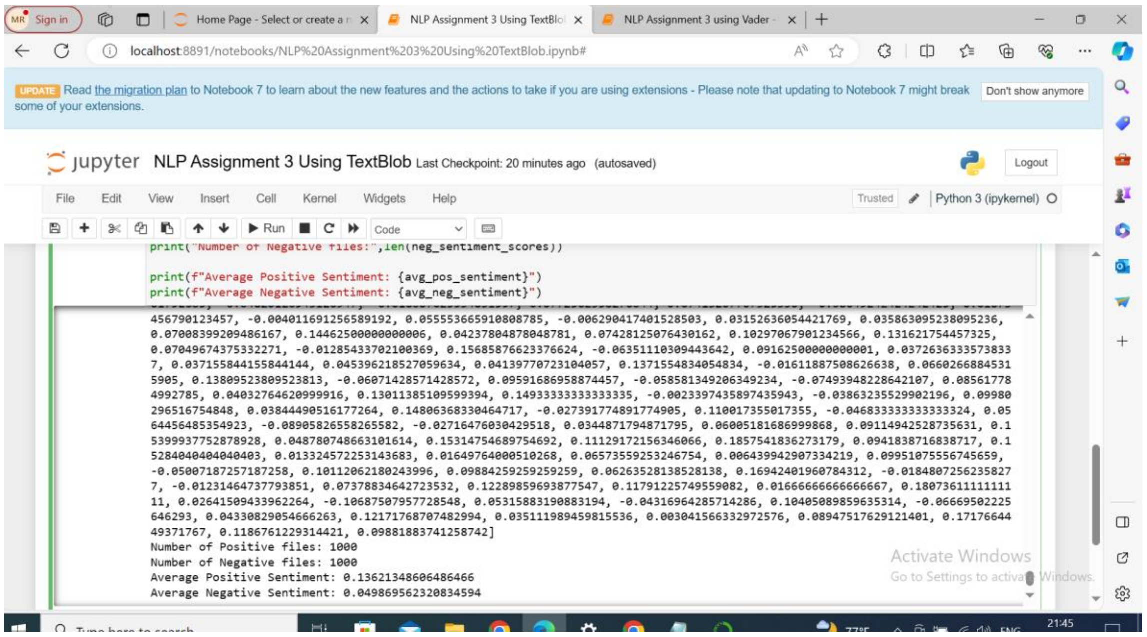
```
In [5]: #Using TextBlob
        from textblob import TextBlob
        import os

        # Paths to your positive and negative text files
        positive_directory = "C://Users//dell//Desktop//Univ docs//Fau Univ docs//2nd Sem FAU//NLP//txt_sentoken//pos"
        negative_directory = "C://Users//dell//Desktop//Univ docs//Fau Univ docs//2nd Sem FAU//NLP//txt_sentoken//neg"

        # To keep track of sentiment scores
        pos_sentiment_scores = []
        neg_sentiment_scores = []

        # Function to analyze sentiment in a text file using TextBlob
        def analyze_textblob_sentiment(text):
            analysis = TextBlob(text)
            return analysis.sentiment.polarity

        # Loop through positive text files
        for filename in os.listdir(positive_directory):
            if filename.endswith(".txt"):
                with open(os.path.join(positive_directory, filename), "r") as file:
```



NLP Assignment 3 Using TextBlob Last Checkpoint: 18 minutes ago (autosaved)

```
        for filename in os.listdir(negative_directory):
            if filename.endswith(".txt"):
                with open(os.path.join(negative_directory, filename), "r") as file:
                    text = file.read()
                    sentiment_score = analyze_textblob_sentiment(text)
                    neg_sentiment_scores.append(sentiment_score)

        avg_pos_sentiment = sum(pos_sentiment_scores) / len(pos_sentiment_scores)
        avg_neg_sentiment = sum(neg_sentiment_scores) / len(neg_sentiment_scores)

        print("Sentiment scores of 1000 positive files",pos_sentiment_scores)
        print("\nSentiment scores of 1000 negative files",neg_sentiment_scores)

        print("Number of Positive files:",len(pos_sentiment_scores))
        print("Number of Negative files:",len(neg_sentiment_scores))

        print(f"Average Positive Sentiment: {avg_pos_sentiment}")
        print(f"Average Negative Sentiment: {avg_neg_sentiment}")
```

```
Sentiment scores of 1000 positive files [0.02366334963737561, 0.103847233495671, 0.13109217171717172, 0.1106257682646571, -0.
07015070346320346, 0.12210534686341136, 0.11495892100730809, 0.11924991674991678, -0.12719298245614036, 0.11019480519480518,
0.04228752728752727, 0.2019753086419753, 0.2447301136363636, 0.028888888888888898, 0.17452976190476185, 0.1346582832590295,
0.2144774531024531, 0.20394303902116404, 0.2113172134882661, 0.11251909854851032, 0.1466119528619529, 0.08307501441647784, 0.
17826253607503606, 0.11705687830687832, 0.07031989919313864, 0.21057773109243694, 0.14750258799171842, 0.2216212593990371 5,
```


```
print("Number of Negative files:",len(neg_sentiment_scores))

print(f"Average Positive Sentiment: {avg_pos_sentiment}")
print(f"Average Negative Sentiment: {avg_neg_sentiment}")
```

```
456790123457, -0.004011691256589192, 0.055553665910808785, -0.006290417401528503, 0.03152636054421769, 0.035863095238095236,
0.07008399209486167, 0.14462500000000006, 0.04237804878048781, 0.07428125076430162, 0.10297067901234566, 0.131621754457325,
0.07049674375332271, -0.01285433702100369, 0.15685876623376624, -0.06351110309443642, 0.09162500000000001, 0.0372636333573833
7, 0.037155844155844144, 0.045396218527059634, 0.04139770723104057, 0.1371554834054834, -0.01611887508626638, 0.0660266884531
5905, 0.13809523809523813, -0.06071428571428572, 0.09591686958874457, -0.05858134920634923, -0.07493948228642107, 0.08561778
4992785, 0.04032764620999916, 0.13011385109599394, 0.14933333333333335, -0.0023397435897435943, -0.03863235529902196, 0.09980
296516754848, 0.03844490516177264, 0.14806368330464717, -0.027391774891774905, 0.110017355017355, -0.04683333333333324, 0.05
64456485354923, -0.08905826558265582, -0.02716476030429518, 0.0344871794871795, 0.06005181686999868, 0.09114942528735631, 0.1
5399937752878928, 0.048780748663101614, 0.15314754689754692, 0.11129172156346066, 0.1857541836273179, 0.0941838716838717, 0.1
5284040404040403, 0.013324572253143683, 0.01649764000510268, 0.06573559253246754, 0.006439942907334219, 0.09951075556745659,
-0.05007187257187258, 0.10112062180243996, 0.09884259259259259, 0.06263528138528138, 0.16942401960784312, -0.018480725623582
7, -0.01231464737793851, 0.07378834642723532, 0.12289859693877547, 0.11791225749559082, 0.016666666666666667, 0.18073611111111
11, 0.02641509433962264, -0.10687507957728548, 0.05315883190883194, -0.04316964285714286, 0.10405089859635314, -0.06669502225
646293, 0.0433082905466263, 0.12171768707482994, 0.035111989459815536, 0.003041566332972576, 0.08947517629121401, 0.17176644
49371767, 0.1186761229314421, 0.09881883741258742]
Number of Positive files: 1000
Number of Negative files: 1000
Average Positive Sentiment: 0.13621348606486466
Average Negative Sentiment: 0.049869562320834594
```

<u>Using Vader:</u>

<u>Code:</u>

#Using Vader

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

import os

# Paths to your positive and negative text files

positive_directory = "C://Users//dell//Desktop//Univ docs//Fau Univ docs//2nd Sem FAU//NLP//txt_sentoken//pos"

negative_directory = "C://Users//dell//Desktop//Univ docs//Fau Univ docs//2nd Sem FAU//NLP//txt_sentoken//neg"

vader_analyzer = SentimentIntensityAnalyzer()

# To keep track of sentiment scores

pos_sentiment_scores = []

neg_sentiment_scores = []

# Function to analyze sentiment in a text file using VADER

def analyze_vader_sentiment(text):

  sentiment = vader_analyzer.polarity_scores(text)

  return sentiment['compound']

```
# Loop through positive text files
for filename in os.listdir(positive_directory):
    if filename.endswith(".txt"):
        with open(os.path.join(positive_directory, filename), "r") as file:
            text = file.read()
            sentiment_score = analyze_vader_sentiment(text)
            pos_sentiment_scores.append(sentiment_score)
# Loop through negative text files
for filename in os.listdir(negative_directory):
    if filename.endswith(".txt"):
        with open(os.path.join(negative_directory, filename), "r") as file:
            text = file.read()
            sentiment_score = analyze_vader_sentiment(text)
            neg_sentiment_scores.append(sentiment_score)
avg_pos_sentiment = sum(pos_sentiment_scores) / len(pos_sentiment_scores)
avg_neg_sentiment = sum(neg_sentiment_scores) / len(neg_sentiment_scores)
print("Sentiment scores of 1000 positive files",pos_sentiment_scores)
print("\nSentiment scores of 1000 negative files",neg_sentiment_scores)
print("Number of Positive files:",len(pos_sentiment_scores))
print("Number of Negative files:",len(neg_sentiment_scores))
print(f"Average Positive Sentiment: {avg_pos_sentiment}")
print(f"Average Negative Sentiment: {avg_neg_sentiment}")
```

Code Explanation:

Data Source:

- The code analyzes two sets of text files: positive and negative.

Sentiment Analysis Method:

- It employs VADER, which is a tool specialized for sentiment analysis.
- VADER calculates a "compound" score to determine the overall sentiment of a piece of text.

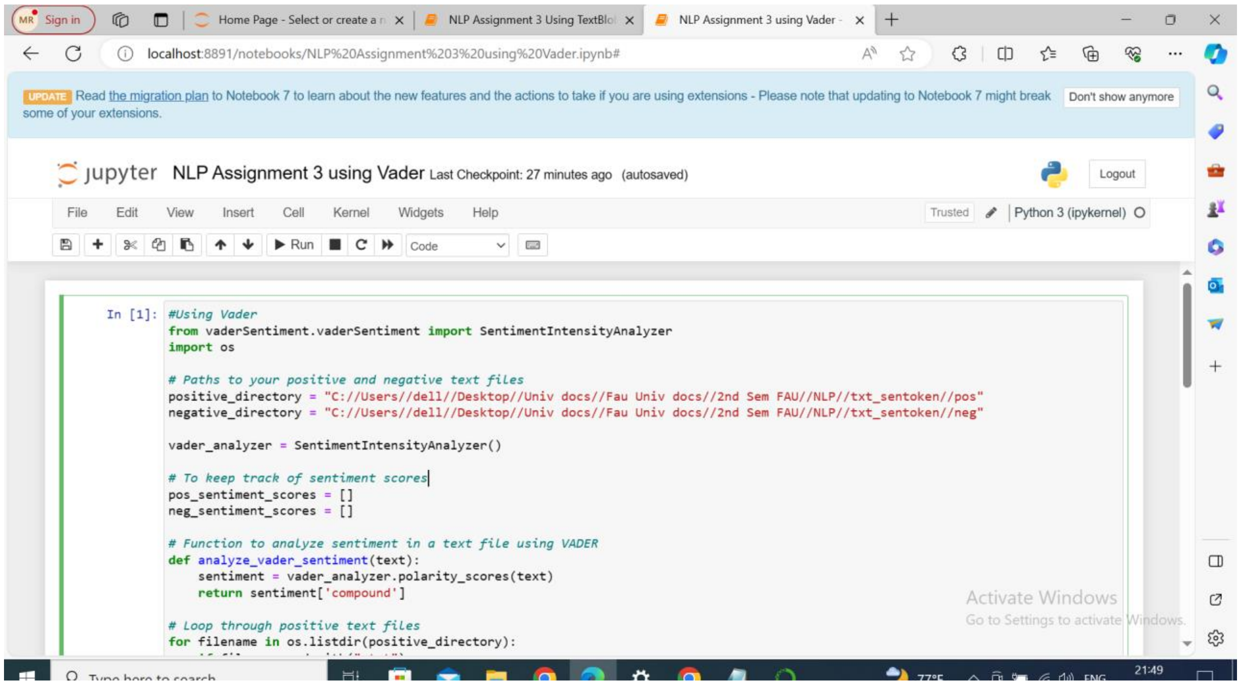
Data Processing:

- The code reads each text file and evaluates its sentiment score using VADER.
- For positive files, it collects the sentiment scores into the **pos_sentiment_scores** list.
- For negative files, it does the same but stores scores in the **neg_sentiment_scores** list.

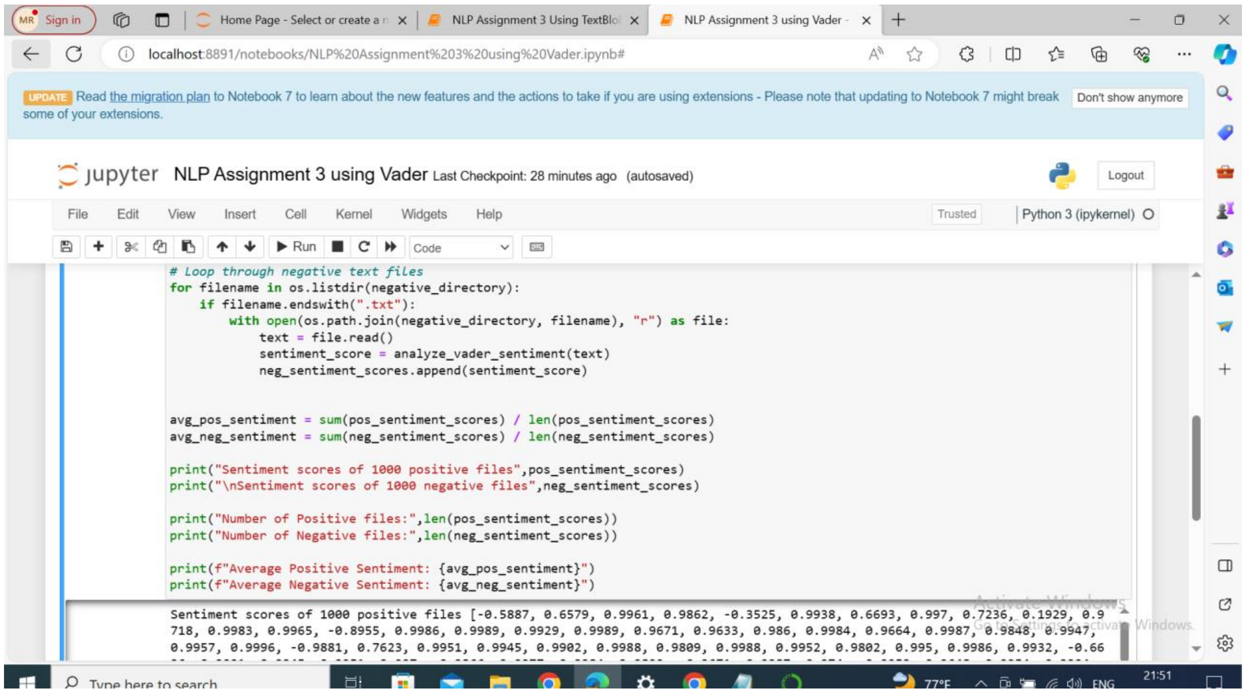
Displaying Results:

- The code provides insight into the sentiment of the text data by revealing:
- The sentiment scores for each individual file.
- The number of files in both positive and negative categories.
- The average sentiment scores for each category.

<u>Output Snapshots:</u>

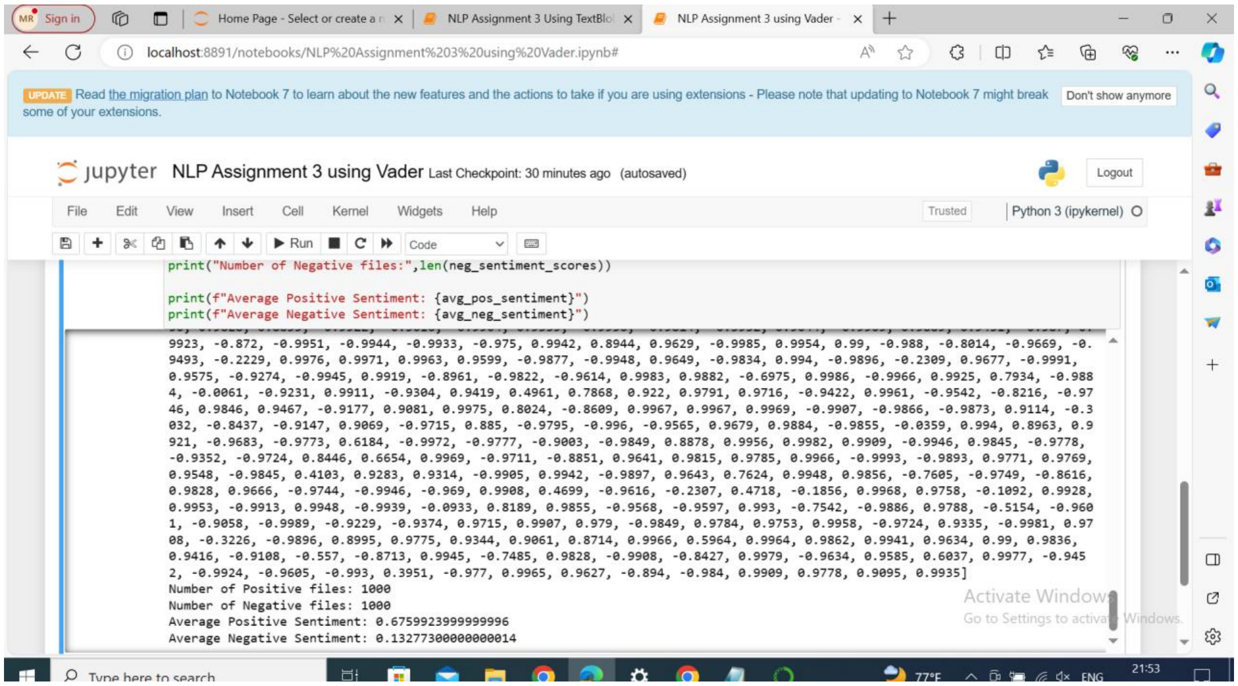

Home Page - Select or create a n
NLP Assignment 3 Using TextBlo
NLP Assignment 3 using Vader -
localhost:8891/notebooks/NLP%20Assignment%203%20using%20Vader.ipynb#
UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
Don't show anymore
jupyter NLP Assignment 3 using Vader Last Checkpoint: 27 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [1]: #Using Vader
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
import os

# Paths to your positive and negative text files
positive_directory = "C://Users//dell//Desktop//Univ docs//Fau Univ docs//2nd Sem FAU//NLP//txt_sentoken//pos"
negative_directory = "C://Users//dell//Desktop//Univ docs//Fau Univ docs//2nd Sem FAU//NLP//txt_sentoken//neg"

vader_analyzer = SentimentIntensityAnalyzer()

# To keep track of sentiment scores
pos_sentiment_scores = []
neg_sentiment_scores = []

# Function to analyze sentiment in a text file using VADER
def analyze_vader_sentiment(text):
    sentiment = vader_analyzer.polarity_scores(text)
    return sentiment['compound']

# Loop through positive text files
for filename in os.listdir(positive_directory):
```



```python
# Loop through negative text files
for filename in os.listdir(negative_directory):
    if filename.endswith(".txt"):
        with open(os.path.join(negative_directory, filename), "r") as file:
            text = file.read()
            sentiment_score = analyze_vader_sentiment(text)
            neg_sentiment_scores.append(sentiment_score)


avg_pos_sentiment = sum(pos_sentiment_scores) / len(pos_sentiment_scores)
avg_neg_sentiment = sum(neg_sentiment_scores) / len(neg_sentiment_scores)

print("Sentiment scores of 1000 positive files",pos_sentiment_scores)
print("\nSentiment scores of 1000 negative files",neg_sentiment_scores)

print("Number of Positive files:",len(pos_sentiment_scores))
print("Number of Negative files:",len(neg_sentiment_scores))

print(f"Average Positive Sentiment: {avg_pos_sentiment}")
print(f"Average Negative Sentiment: {avg_neg_sentiment}")
```

```
Sentiment scores of 1000 positive files [-0.5887, 0.6579, 0.9961, 0.9862, -0.3525, 0.9938, 0.6693, 0.997, 0.7236, 0.1929, 0.9
718, 0.9983, 0.9965, -0.8955, 0.9986, 0.9989, 0.9929, 0.9989, 0.9671, 0.9633, 0.986, 0.9984, 0.9664, 0.9987, 0.9848, 0.9947,
0.9957, 0.9996, -0.9881, 0.7623, 0.9951, 0.9945, 0.9902, 0.9988, 0.9809, 0.9988, 0.9952, 0.9802, 0.995, 0.9986, 0.9932, -0.66
```



```python
print("Number of Negative files:",len(neg_sentiment_scores))

print(f"Average Positive Sentiment: {avg_pos_sentiment}")
print(f"Average Negative Sentiment: {avg_neg_sentiment}")
```

```
9923, -0.872, -0.9951, -0.9944, -0.9933, -0.975, 0.9942, 0.8944, 0.9629, -0.9985, 0.9954, 0.99, -0.988, -0.8014, -0.9669, -0.
9493, -0.2229, 0.9976, 0.9971, 0.9963, 0.9599, -0.9877, -0.9948, 0.9649, -0.9834, 0.994, -0.9896, -0.2309, 0.9677, -0.9991,
0.9575, -0.9274, -0.9945, 0.9919, -0.8961, -0.9822, -0.9614, 0.9983, 0.9882, -0.6975, 0.9986, -0.9966, 0.9925, 0.7934, -0.988
4, -0.0061, -0.9231, 0.9911, -0.9304, 0.9419, 0.4961, 0.7868, 0.922, 0.9791, 0.9716, -0.9422, 0.9961, -0.9542, -0.8216, -0.97
46, 0.9846, 0.9467, -0.9177, 0.9081, 0.9975, 0.8024, -0.8609, 0.9967, 0.9967, 0.9969, -0.9907, -0.9866, -0.9873, 0.9114, -0.3
032, -0.8437, -0.9147, 0.9069, -0.9715, 0.885, -0.9795, -0.996, -0.9565, 0.9679, 0.9884, -0.9855, -0.0359, 0.994, 0.8963, 0.9
921, -0.9683, -0.9773, 0.6184, -0.9972, -0.9777, -0.9003, -0.9849, 0.8878, 0.9956, 0.9982, 0.9909, -0.9946, 0.9845, -0.9778,
-0.9352, -0.9724, 0.8446, 0.6654, 0.9969, -0.9711, -0.8851, 0.9641, 0.9815, 0.9785, 0.9966, -0.9993, -0.9893, 0.9771, 0.9769,
0.9548, -0.9845, 0.4103, 0.9283, 0.9314, -0.9905, 0.9942, -0.9897, 0.9643, 0.7624, 0.9948, 0.9856, -0.7605, -0.9749, -0.8616,
0.9828, 0.9666, -0.9744, -0.9946, -0.969, 0.9908, 0.4699, -0.9616, -0.2307, 0.4718, -0.1856, 0.9968, 0.9758, -0.1092, 0.9928,
0.9953, -0.9913, 0.9948, -0.9939, -0.0933, 0.8189, 0.9855, -0.9568, -0.9597, 0.993, -0.7542, -0.9886, 0.9788, -0.5154, -0.960
1, -0.9058, -0.9989, -0.9229, -0.9374, 0.9715, 0.9907, 0.979, -0.9849, 0.9784, 0.9753, 0.9958, -0.9724, 0.9335, -0.9981, 0.97
08, -0.3226, -0.9896, 0.8995, 0.9775, 0.9344, 0.9061, 0.8714, 0.9966, 0.5964, 0.9964, 0.9862, 0.9941, 0.9634, 0.99, 0.9836,
0.9416, -0.9108, -0.557, -0.8713, 0.9945, -0.7485, 0.9828, -0.9908, -0.8427, 0.9979, -0.9634, 0.9585, 0.6037, 0.9977, -0.945
2, -0.9924, -0.9605, -0.993, 0.3951, -0.977, 0.9965, 0.9627, -0.894, -0.984, 0.9909, 0.9778, 0.9095, 0.9935]
Number of Positive files: 1000
Number of Negative files: 1000
Average Positive Sentiment: 0.6759923999999996
Average Negative Sentiment: 0.13277300000000014
```

Comparison TextBlob and Vader:

Accuracy:

- TextBlob: TextBlob may provide reasonably accurate sentiment analysis results, but it may not always be perfect, especially for nuanced or context-dependent sentiments.
- VADER: VADER is known for its good accuracy, especially in distinguishing between positive, negative, and neutral sentiments. It's well-suited for social media data.

Ease of Use:

- TextBlob: TextBlob is easy to use and provides a simple interface for sentiment analysis. It's a good choice for beginners.
- VADER: VADER is also user-friendly and provides a straightforward way to analyze sentiment without much configuration.

Customization:

- TextBlob: TextBlob allows some degree of customization, such as using custom training data for specific domains or languages.
- VADER: VADER is less customizable, and it's primarily designed for general sentiment analysis.

Speed:

- TextBlob: TextBlob's sentiment analysis might be slower when dealing with a large number of documents due to its linguistic analysis.
- VADER: VADER is relatively faster, which makes it suitable for real-time or large-scale sentiment analysis.

Handling of Sarcasm and Negation:

- TextBlob: TextBlob might struggle with detecting sarcasm or negation in text, potentially misclassifying sentiments.
- VADER: VADER has some ability to handle negation and context, making it more robust in the presence of sarcasm or nuanced language.

Dependencies:

- TextBlob: TextBlob has additional dependencies, including NLTK, which might require more setup and resources.
- VADER: VADER is self-contained and doesn't rely on external dependencies, making it easier to deploy.

Domain-Specific Analysis:

- TextBlob: TextBlob can be fine-tuned for domain-specific sentiment analysis, which is useful for certain industries or applications.
- VADER: VADER is better suited for general sentiment analysis and may require more effort to adapt to specific domains.