SD21063 TEAN JIN HE Data Mining Lab Report 6

January 20, 2024

1 Data Mining Lab Report 6

NAME: TEAN JIN HE

MATRIC ID: SD21063

SECTION: 02G

1.0.1 Case Study:

Text mining, also known as text data mining, is the process of transforming unstructured text into a structured format to identify meaningful patterns and new insights (IBM, 2022). Natural Language Processing (NLP) is a field of AI that allows machines to read, understand and derive meaning from human languages (Diego Lopez Yse, 2019). There are several concepts to understand NLP, such as tokenization, stemming and lemmatization to process the text.

1.0.2 Question 1

General Knowledge

Discuss the text mining concepts in data mining implementation related to sentiment analysis applications discussed above. Give reference/ references. The text mining concepts in data mining implementation related to sentiment analysis applications are:

- Voice of the customer analysis: This concept refers to the systematic examination of consumer input in order to get practical and implementable insights. These insights provide a deeper comprehension of client behaviour and enable the execution of marketing-conversion gap analysis.
- Social media listening: This concept refers to the process of monitoring and analyzing the online conversations and sentiments of users on various social media platforms. This can help businesses understand their customers' preferences, opinions, and feedback, as well as identify trends and opportunities.
- Sentiment analysis: This concept refers to the application of natural language processing, text analysis, and computational linguistics to identify and extract the subjective information and emotional states of the text. This can help businesses measure the attitudes, emotions, and opinions of their customers and stakeholders.

Reference:
https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-

6c94d6f58c17 https://www.repustate.com/blog/text-sentiment-analysis/ https://www.red-gate.com/simple-talk/databases/sql-server/bi-sql-server/text-mining-and-sentiment-analysis-introduction/

1.0.3 Question 2

Python

1.0.4 Based on the aforementioned above, develop sentiment analyser for movie review with criteria as follows:

1.0.5 a. Import movie_reviews corpus from nltk.corpus.

[1]:
```python
import nltk
from nltk import FreqDist
from nltk.corpus import movie_reviews
from nltk.classify import NaiveBayesClassifier
from nltk.classify.util import accuracy
```

[2]:
```python
# Download the movie_reviews corpus if not already downloaded
nltk.download('movie_reviews')
```

```
[nltk_data] Downloading package movie_reviews to
[nltk_data]     C:\Users\user\AppData\Roaming\nltk_data…
[nltk_data]   Package movie_reviews is already up-to-date!
```

[2]:
```
True
```

1.0.6 b. Treshold setting = 0.9.

[3]:
```python
# Extract features from the input list of words
def extract_features(words):
    return dict([(word, True) for word in words])

if __name__=='__main__':
    # Load the reviews from the corpus
    fileids_pos = movie_reviews.fileids('pos')
    fileids_neg = movie_reviews.fileids('neg')

    # Extract the features from the reviews
    features_pos = [(extract_features(movie_reviews.words(
        fileids=[f])), 'Positive') for f in fileids_pos]
    features_neg = [(extract_features(movie_reviews.words(
        fileids=[f])), 'Negative') for f in fileids_neg]

    # Define the train and test split (80% and 20%)
    threshold = 0.9
    num_pos = int(threshold * len(features_pos))
    num_neg = int(threshold * len(features_neg))

    # Create training and test datasets
    training_features = features_pos[:num_pos] + features_neg[:num_neg]
    testing_features = features_pos[num_pos:] + features_neg[num_neg:]

    # Print the number of datapoints used
    print('\nNumber of training datapoints:', len(training_features))
    print('Number of test datapoints:', len(testing_features))
```

```
Number of training datapoints: 1800
Number of test datapoints: 200
```

1.0.7 c. Use Naïve Bayes classifier for model development and set the accuracy result.

[4]:
```python
# Train the Naïve Bayes classifier
classifier = NaiveBayesClassifier.train(training_features)

# Evaluate the accuracy of the classifier on the testing set
accuracy_result = accuracy(classifier, testing_features)
print(f'Accuracy og the classifier: {accuracy_result:.2%}')
```

```
Accuracy og the classifier: 73.00%
```

1.0.8 d. Create 20 most informative words.

[5]:
```python
N = 20
print('\nTop ' + str(N) + ' Most Informative Words:')
for i, item in enumerate(classifier.most_informative_features()):
    print(str(i+1) + '. ' + item[0])
    if i == N - 1:
        break
```

```
Top 20 Most Informative Words:
1. outstanding
2. ludicrous
3. astounding
4. avoids
5. idiotic
6. atrocious
7. fascination
8. offbeat
9. animators
10. symbol
11. religion
12. annual
13. hudson
14. jolie
```

```
15. represent
16. insulting
17. addresses
18. dread
19. fairness
20. hatred
```

1.0.9 e. Test the input movie reviews with the reviews below:

i. The costumes in this movie look real.

ii. The movie is merely based on the book.

iii. I think the story was terrible and the characters were very weak.

iv. People say that the director of the movie is astounding.

v. This is such a ridiculous movie. I will not recommend it to anyone.

[6]:
```python
# Test input movie reviews
input_reviews = [
    "The costumes in this movie look real.",
    "The movie is merely based on the book.",
    "I think the story was terrible and the characters were very weak.",
    "People say that the director of the movie is astounding.",
    "This is such a ridiculous movie. I will not recommend it to anyone."
]
```

1.0.10 f. Predict the sentiment and probability of reviews in (e).

[7]:
```python
# Predict sentiment and probability for each input review
print("\nMovie review predictions:")
for review in input_reviews:
    print("\nReview:", review)

    # Compute the probabilities
    probabilities = classifier.prob_classify(extract_features(review.split()))

    # Pick the maximum value
    predicted_sentiment = probabilities.max()

    # Print outputs
    print("Predicted sentiment:", predicted_sentiment)
    print(f"Probability: {probabilities.prob(predicted_sentiment):.2%}")
```

```
Movie review predictions:
```

```
Review: The costumes in this movie look real.
Predicted sentiment: Positive
Probability: 50.52%

Review: The movie is merely based on the book.
Predicted sentiment: Positive
Probability: 56.29%

Review: I think the story was terrible and the characters were very weak.
Predicted sentiment: Negative
Probability: 72.44%

Review: People say that the director of the movie is astounding.
Predicted sentiment: Negative
Probability: 54.72%

Review: This is such a ridiculous movie. I will not recommend it to anyone.
Predicted sentiment: Negative
Probability: 78.38%
```

[ ]: