To,

IITD-AIA Foundation on Smart Manufacturing

Subject: Weekly Progress Report for Week-8

Respected sir,

Following is the required progress report to the best of my knowledge considering relevant topics to be covered:

- Silhouette coefficient .
- Tokenizing words and sentences.
- Stop words customization.
- Stemming and Lemmatization .
- Feature extraction .
- Count Vectorizer
- TfidfVectorizer

Day :24

Silhouette coefficient .

The Silhouette coefficient is a measure used to evaluate how well data points fit into their assigned clusters. It ranges from -1 to 1, with higher values indicating better-defined clusters. A value close to +1 means well-clustered data, while a value close to -1 suggests that a data point may be in the wrong cluster.

Steps:

1. For each data point, calculate its average distance to all other points in the same cluster and find the nearest neighboring cluster.

2. Calculate the average distance from the data point to all points in the neighboring cluster (inter-cluster distance).

3. Compute the Silhouette coefficient for the data point using the formula: (inter-cluster distance - intra-cluster distance) / max(intra-cluster distance, inter-cluster distance).

4. Repeat steps 1 to 3 for all data points to get individual Silhouette coefficients.

5 Calculate the average Silhouette coefficient for all data points to evaluate the overall clustering quality. Higher values indicate better-defined clusters.

Reference: https://towardsdatascience.com/silhouette-coefficient-validating-clustering-techniques-e976bb81d10c

Day: 25

Tokenizing words and sentences.

Tokenizing is the process of breaking text into smaller units, like words or sentences, for analysis in natural language processing (NLP). Word tokenization splits text into individual words, and sentence tokenization divides text into separate sentences.

Tokenizing Words:

1. Import the necessary NLP library
2. Load or input the text you want to tokenize.
3. Use the library's function for word tokenization to split the text into individual words.
4. The output will be a list of words.

Tokenizing Sentences:

1. Import the required NLP library (e.g., NLTK or spaCy).
2. Load or input the text you want to tokenize.
3. Use the library's function for sentence tokenization to divide the text into separate sentences.
4. The output will be a list of sentences.

References :

1. https://www.tokenizer.cc/
2. https://www.geeksforgeeks.org/nlp-how-tokenizing-text-sentence-words-works/

Day:26

Stop words customization .

Stop words customization means modifying the list of common words in text analysis to improve the accuracy and relevance of results for specific tasks or domains. This involves adding or removing words from the default list. It helps to extract more meaningful information from textual data.

Steps:

1. Identify domain or specific context.
2. Create a default list of stop words.
3. Analyze the textual data to identify domain-specific words and common meaningless words.
4. Customize the stop words list by adding domain-specific words and removing general stop words.
5. Fine-tune the stop words list through iterative testing.
6. Implement the customized stop words list into your natural language processing pipeline.
7. Evaluate the performance of your results to ensure the customization improves output.
8. Iterate if necessary, refining the stop words list based on evaluation.

Algorithm:
Input:

- Textual data for analysis

- Default list of stop words

- Domain-specific words (optional)

Output:

Customized list of stop words

Algorithm:

1. Initialize an empty set to store customized stop words.

2. Analyze the textual data to identify frequently occurring words.

3. Compare the frequently occurring words with the default stop words list.

4. Add the domain-specific words (if provided) to the set of customized stop words.

5. Remove any general stop words from the set of customized stop words that are important in your domain.

6. Output the customized stop words list.

Reference:

https://www.youtube.com/watch?v=vUPAOU2NPIs&pp=ygUYc3RvcCB3b3JkcyBjdXN0b21pemF0aW9u

Day:27

Stemming and Lemmatization .

- Stemming: Faster, less accurate. Reduces words to their root form, but the result may not be a valid word.
- Lemmatization: Slower, more accurate. Reduces words to their base form , which is a valid word in the dictionary and considers grammatical properties.

Steps involved :

Stemming:

1. Tokenize the text into words.
2. Apply a stemming algorithm to obtain word stems .
3. Less accurate.

Lemmatization:

1. Tokenize the text into words.
2. Perform part-of-speech tagging.
3. Apply a lemmatization algorithm to obtain valid word lemmas.
4. More accurate.

References:

https://www.kaggle.com/discussions/general/202461

https://www.youtube.com/watch?v=HHAilAC3cXw&pp=ygUac3RlbW1pbmcgYW5kIGxlbW1hdGl6YXRpb24%3D

Day: 28

Feature extraction .

It involves selecting the most important aspects of the data and creating new useful information from it,

improving the efficiency and accuracy of machine learning algorithms.

Steps involved

1. Prepare and preprocess the data.
2. Rank or score the features based on their importance.
3. Select the top-ranked features or use a threshold to choose relevant ones.
4. Evaluate the model's performance using only the selected features.
5. If needed, iterate and refine the feature selection process.
6. Train and validate the model using the final set of selected features.

Required algorithm for this .

1. Train a machine learning model on the entire set of features.
2. Evaluate the importance of each feature based on the model's performance.
3. Remove the least important feature.
4. Retrain the model using the remaining features.

Reference:

https://www.kaggle.com/code/pmarcelino/data-analysis-and-feature-extraction-with-python

Day: 29

Count Vectorizer.

It is a part of the feature extraction process used to convert text data into numerical vectors, which machine learning algorithms can then work with.

Steps for Count Vectorizer are:

1. Import the library.
2. Create an instance of CountVectorizer.
3. Fit the CountVectorizer to your text data.
4. Transform the text data into a Document-Term Matrix (DTM).
5. Optionally, convert the DTM into a dense matrix.
6. Utilize the DTM for text analysis or machine learning tasks.

1. Input: Text documents
2. Tokenization: Split into words
3. Vocabulary: Unique word list
4. Count: Word frequencies
5. DTM: Document-Term Matrix
6. Sparse Matrix: Often sparse
7. Optional: Dense Matrix
8. Output: DTM for analysis or ML.

References:https://youtu.be/GMIltOKCOjA

Day :30

TfidfVectorizer .

TF-IDF is a text processing technique used in natural language processing and information retrieval. It aims to represent the importance of words in a document relative to a collection of documents.

Steps:

1. Preprocess the document (remove special characters, lowercase, tokenize).
2. Calculate the Term Frequency (TF) for each word in the document.
3. Calculate the Inverse Document Frequency (IDF) for each word in the collection.
4. Multiply TF by IDF to get the TF-IDF score for each word in the document.
5. Represent the document as a TF-IDF vector for further analysis.

example:

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Step 1: Preprocess the document
document = "This is a sample document for TF-IDF calculation."
document = document.lower()  # Convert to lowercase
corpus = [document]  # Create the corpus (collection of documents)

# Step 2: Calculate TF-IDF using TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(corpus)
```

```
# Step 3: Get the TF-IDF vector for the input document
tfidf_vector = tfidf_matrix[0]  # The first (and only) row in the TF-IDF matrix

# Convert the TF-IDF vector to a dense array (optional)
tfidf_vector = tfidf_vector.toarray()

print("TF-IDF Vector:")
print(tfidf_vector)
```

Reference:

http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

https://www.youtube.com/watch?v=D2V1okCEsiE&pp=ygUPVGZpZGZWZWN0b3JpemVy