Text Similarity Using TF-IDF and Cosine Similarity

Overview

In Natural Language Processing (NLP), measuring the similarity between texts is crucial for tasks such as document clustering, information retrieval, and semantic analysis. One effective method combines Term Frequency-Inverse Document Frequency (TF-IDF) vectorization with Cosine Similarity to assess how closely related two pieces of text are.

Why Use TF-IDF and Cosine Similarity?

- **TF-IDF Vectorization**: Transforms textual data into numerical vectors by evaluating the importance of words in a document relative to a collection of documents (corpus). This method reduces the impact of commonly occurring words and highlights unique terms.

- **Cosine Similarity**: Measures the cosine of the angle between two non-zero vectors, providing a metric that quantifies the similarity between them. It is particularly effective for text analysis as it accounts for the direction of the vectors rather than their magnitude.

Prerequisites

- Python 3.x
- Libraries:
    - `scikit-learn`

Files Included

- `text_similarity.py`: Contains the implementation of TF-IDF vectorization and Cosine Similarity calculation.

Code Description

The following code demonstrates how to compute the similarity between two texts using TF-IDF vectorization and Cosine Similarity:

```
 from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

# Sample texts
text1 = "Machine learning is amazing."
text2 = "Deep learning is a subset of machine learning."

# Initialize the TF-IDF Vectorizer
vectorizer = TfidfVectorizer()

# Fit and transform the texts into TF-IDF vectors
tfidf_matrix = vectorizer.fit_transform([text1, text2])

# Compute the Cosine Similarity between the two vectors
similarity = cosine_similarity(tfidf_matrix[0:1], tfidf_matrix[1:2])
```

```
print(f"Cosine Similarity: {similarity[0][0]:.4f}")
```

Explanation:

1. **Import Libraries**:
   - `TfidfVectorizer` from `sklearn.feature_extraction.text` is used to convert the text data into TF-IDF vectors.
   - `cosine_similarity` from `sklearn.metrics.pairwise` calculates the cosine similarity between vectors.
2. **Sample Texts**:
   - Two sample sentences are defined:
     - `text1`: "Machine learning is amazing."
     - `text2`: "Deep learning is a subset of machine learning."
3. **TF-IDF Vectorization**:
   - An instance of `TfidfVectorizer` is created.
   - The `fit_transform` method is applied to the list containing `text1` and `text2`, resulting in a TF-IDF matrix where each row corresponds to a text and each column corresponds to a term's TF-IDF weight.
4. **Cosine Similarity Calculation**:
   - The `cosine_similarity` function computes the similarity between the TF-IDF vectors of `text1` and `text2`.
   - The result is a similarity score between 0 and 1, where 1 indicates identical texts and 0 indicates no similarity.

Expected Output

Running the code will output the cosine similarity score between the two sample texts:

```
Cosine Similarity: 0.6696
```

This score suggests a moderate to high similarity between the two sentences, reflecting their shared focus on "machine learning."

Use Cases

- **Document Clustering**: Grouping similar documents together based on content.
- **Plagiarism Detection**: Identifying copied or closely paraphrased content.
- **Information Retrieval**: Enhancing search engines to retrieve documents relevant to a user's query.

Advantages

- **Efficiency**: TF-IDF combined with Cosine Similarity provides a computationally efficient method for text comparison.
- **Simplicity**: Easy to implement with readily available libraries.
- **Effectiveness**: Produces meaningful similarity scores that align with human intuition.

Future Enhancements

- **Incorporate Synonym Recognition**: Enhance the model to account for synonyms, improving similarity detection between texts with different wording but similar meanings.
- **Use Advanced Embeddings**: Implement word embeddings like Word2Vec or BERT for capturing deeper semantic relationships.
- **Expand to Multi-Document Comparison**: Extend the code to handle and compare multiple documents simultaneously.

References

- TF-IDF and Cosine Similarity in Machine Learning
- Cosine Similarity - Wikipedia
- Python: tf-idf-cosine: to find document similarity - Stack Overflow

---