Assignment: Word Embeddings and Similarity Measures

Objective:

The goal of this assignment is to introduce students to the concept of word embeddings and familiarize them with different similarity measures using Python. Students will generate word embeddings for a set of documents and then employ various similarity metrics to measure the similarity between these documents.

Task 1: Word Embeddings

1.1 Background

Word embeddings are numerical representations of words that capture semantic relationships and contextual information. They allow us to represent words in a continuous vector space, where words with similar meanings are closer together. One popular method for generating word embeddings is through models like Word2Vec or GloVe, which are trained on large text corpora.

1.2 Tokenization

Before generating word embeddings, it is essential to preprocess the text. Tokenization is the process of breaking down text into individual words or tokens. This step is crucial because it converts a continuous text into discrete units that can be fed into the word embedding model. Tokenization helps in creating a structured input for the model and ensures that the semantic meaning of words is preserved.

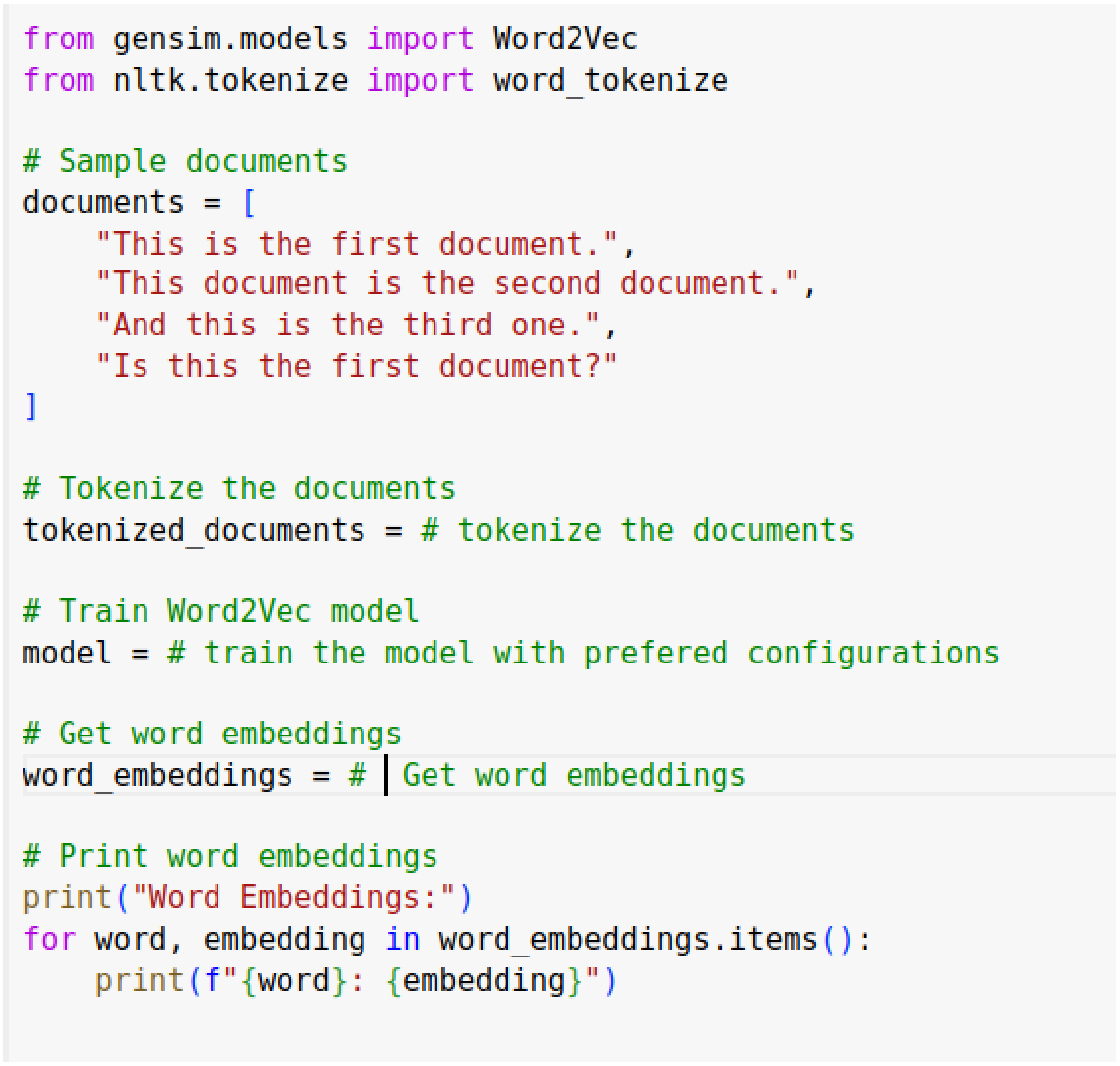
1.2 Implementation

1.2.1 Tokenization

For this task, you can use the `nltk` library in Python for tokenization or any other preferred library

1.2.1 Word embedding

For this task, you can use the `gensim(recommended)`, FastText or WordEmbed libraries in Python to generate word embeddings or any other preferred library/code. They provide an easy-to-use interface for Word2Vec models.



```python
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize

# Sample documents
documents = [
    "This is the first document.",
    "This document is the second document.",
    "And this is the third one.",
    "Is this the first document?"
]

# Tokenize the documents
tokenized_documents = # tokenize the documents

# Train Word2Vec model
model = # train the model with prefered configurations

# Get word embeddings
word_embeddings = # Get word embeddings

# Print word embeddings
print("Word Embeddings:")
for word, embedding in word_embeddings.items():
    print(f"{word}: {embedding}")
```

Task 2: Document Similarity Measures

2.1 Background

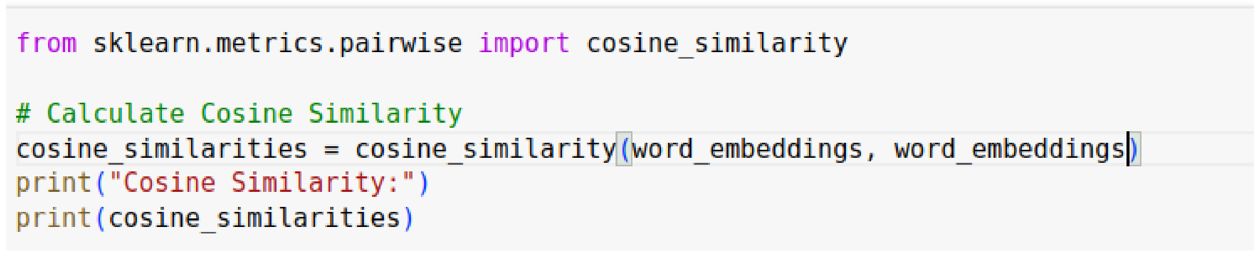
Document similarity measures quantify the degree of similarity between two documents based on their word embeddings. These measures play a crucial role in natural language processing tasks such as document clustering, information retrieval, and recommendation systems. Three common similarity measures are Cosine Similarity, Jaccard Similarity, and Euclidean Distance

2.2 Similarity Measures

- **Cosine Similarity:** Measures the cosine of the angle between two vectors. It is widely used for comparing documents represented as vectors in a high-dimensional space. Cosine Similarity ranges from -1 (completely dissimilar) to 1 (completely similar), with 0 indicating orthogonality.
- **Jaccard Similarity:** Computes the intersection over the union of two sets. In the context of document similarity, the sets are the sets of words present in each document. Jaccard Similarity ranges from 0 to 1, with 1 indicating identical sets.
- **Euclidean Distance:** Measures the straight-line distance between two points in space. In the context of document similarity, it quantifies how far apart the documents are in the vector space. Smaller distances indicate greater similarity.

2.3 Implementation

In this task, you are required to calculate document similarity using these similarity measures based on the word embeddings generated in Task 1. Each measure provides a different perspective on similarity, capturing aspects such as direction, overlap, and spatial distance in the vector space.



```python
from sklearn.metrics.pairwise import cosine_similarity

# Calculate Cosine Similarity
cosine_similarities = cosine_similarity(word_embeddings, word_embeddings)
print("Cosine Similarity:")
print(cosine_similarities)
```

Submission Guidelines:

1. Provide a notebook containing the code for word embeddings generation and document similarity calculation.
2. Include in the notebook the following analysis
    a. Analyzing the results obtained from different similarity measures, discuss the strengths and limitations of each

Instructions

- The assignment is team of 3
- All of the team **MUST** understand all of the code
- You must understand every part in the code

- **Bonus**:
    - Visualization of document similarities provides a clear understanding of relationships between different texts. Heatmaps and multidimensional scaling (MDS) are effective visualization techniques. Visualize the documents similarities using these 2 visualization techniques