Name: Satyatma chincholi

USN: 1RVU23CSE414

Date:27-01-2026

LAB-3 Report: Implementation and Visualization of Word2Vec Embeddings

This lab demonstrates the process of generating semantic word embeddings from a text corpus using the Word2Vec algorithm.

The objective is to represent words as dense vectors in a continuous vector space where semantically similar words are positioned close to one another.

The implementation utilizes several key Python libraries for natural language processing and data science:

- **Gensim**: Used for training the Word2Vec model and managing word vectors.
- **NLTK**: Employed for text preprocessing and tokenization.
- **Scikit-Learn**: Utilized for Principal Component Analysis (PCA) to reduce vector dimensions.
- **Matplotlib**: Used to generate the final 2D visualization of the word space.

The input data consists of a sample corpus focusing on Natural Language Processing (NLP) concepts. Before training, the text undergoes the following transformations:

Normalization: Converting all text to lowercase to ensure consistency.

Tokenization: Utilizing NLTK's word_tokenize to split sentences into individual word units (tokens).

```python
!pip install gensim scikit-learn matplotlib
from gensim.models import Word2Vec
from nltk.tokenize import word_tokenize
import nltk

nltk.download('punkt')
nltk.download('punkt_tab') # Added to download the missing resource

# Sample corpus
corpus = [
    "Natural language processing is a fascinating field",
    "Word embeddings capture semantic meanings",
    "NLP is used in chatbots and virtual assistants",
    "Word2Vec is a powerful tool for creating word embeddings"
]

# Tokenize sentences
tokenized_corpus = [word_tokenize(sentence.lower()) for sentence in corpus]
print(tokenized_corpus)

# Train Word2Vec model
model = Word2Vec(sentences=tokenized_corpus, vector_size=100, window=5, min_count=1, workers=4)

# Save the model
model.save("word2vec.model")

model = Word2Vec.load("word2vec.model")

# Get vectors for a subset of words
words = list(model.wv.index_to_key)[:10]  # Select the first 10 words
print(words)
word_vectors = [model.wv[word] for word in words]
print(word_vectors)
```

Because 100-dimensional vectors cannot be visualized directly, the project applies **Principal Component Analysis (PCA)**. This mathematical technique reduces the vectors to 2 components while preserving as much variance as possible, allowing the words to be plotted on a 2D grid.

```python
from sklearn.decomposition import PCA

# Apply PCA for dimensionality reduction
pca = PCA(n_components=2)
pca_result = pca.fit_transform(word_vectors)

import matplotlib.pyplot as plt

# Plot the words in 2D space
plt.figure(figsize=(10, 5))
plt.scatter(pca_result[:, 0], pca_result[:, 1])

# Annotate the points with the words
for i, word in enumerate(words):
    plt.annotate(word, xy=(pca_result[i, 0], pca_result[i, 1]))

plt.title("2D Visualization of Word Embeddings")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.grid(True)
plt.show()
```

Results and Analysis

The output demonstrates the model's ability to extract specific vectors for tokens like "is," "embeddings," and "word2vec".

- **Vector Output**: The model generates precise floating-point arrays for every word in the vocabulary.
- **Graphical Representation**: The PCA plot visualizes the spatial relationship between terms, providing a map of how the model has learned the "meaning" of the input text.

```
Installing collected packages: gensim
Successfully installed gensim-4.4.0
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[['natural', 'language', 'processing', 'is', 'a', 'fascinating', 'field'], ['word', 'embeddings', 'capture', 'semantic', 'meanings'], ['nlp', 'is', 'used', 'in', 'chatbots', 'and', 'virtual', 'assistants'], ['word2vec', 'is', 'a', 'powerfu
['is', 'embeddings', 'word', 'a', 'creating', 'for', 'tool', 'powerful', 'word2vec', 'assistants']
[array([-5.3482049e-04,  2.3825991e-04,  5.1056668e-03,  9.0133334e-03,
       -9.3041826e-03, -7.1184216e-03,  6.4591165e-03,  8.9739896e-03,
       -5.0175558e-03, -3.7653942e-03,  7.3820921e-03, -1.5338163e-03,
       -4.5345500e-03,  6.5530594e-03, -4.8587834e-03, -1.8163866e-03,
        2.8779216e-03,  9.9028728e-04, -8.2870843e-03, -9.4514126e-03,
        7.3109688e-03,  5.0709019e-03,  6.7580235e-03,  7.6032913e-04,
        6.3481932e-03, -3.4047901e-03, -9.4892143e-04,  5.7721287e-03,
       -7.5215534e-03, -3.9352756e-03, -7.5127776e-03, -9.3112560e-04,
        9.5384931e-03, -7.3194429e-03, -2.3347498e-03, -1.9366210e-03,
        8.0779372e-03, -5.9318673e-03,  4.1685435e-05, -4.7525410e-03,
       -9.6020838e-03,  5.0074058e-03, -8.7600304e-03, -4.3903729e-03,
       -3.4098470e-05, -2.9744051e-04, -7.6616611e-03,  9.6138855e-03,
        4.9824370e-03,  9.2338668e-03, -8.1582749e-03,  4.4954196e-03,
       -4.1358816e-03,  8.2416082e-04,  8.4963180e-03, -4.4610659e-03,
        4.5164647e-03, -6.7853210e-03, -3.5473362e-03,  9.3990080e-03,
       -1.5771524e-03,  3.2147145e-04, -4.1403268e-03, -7.6813265e-03,
       -1.5084570e-03,  2.4713580e-03, -8.8901544e-04,  5.5368468e-03,
       -2.7449303e-03,  2.2591040e-03,  5.4555875e-03,  8.3476352e-03,
       -1.4524100e-03, -9.2081940e-03,  4.3700617e-03,  5.7310116e-04,
        7.4422611e-03, -8.1312557e-04, -2.6384799e-03, -8.7542878e-03,
       -8.5941557e-04,  2.8278236e-03,  5.4004234e-03,  7.0539773e-03,
```