Experiment 3: Train a custom Word2Vec model on a small dataset. Train embedding's on a domain-specific corpus (e.g., legal, medical) and analyse how embedding's capture domain-specific semantics.

Experiment 3: Word2Vec Training on Medical Corpus with Bigram Detection

Introduction

Word embeddings are numerical vector representations of words that capture their semantic meaning. **Word2Vec**, introduced by Mikolov et al., is a popular algorithm for learning such embeddings. It is used in **Natural Language Processing (NLP)** to represent words in a continuous vector space, where similar words have similar representations.

In this experiment, we train a **custom Word2Vec model** on a **medical corpus** to generate meaningful word embeddings. To improve the quality of embeddings, **bigrams (multi-word phrases)** are detected and incorporated into the model. The trained model is then analyzed by checking similar words and visualizing word relationships using **Principal Component Analysis (PCA).**

Objectives

1. **Train a Word2Vec model** on a **medical domain-specific** text corpus.
2. **Remove stopwords** to improve the quality of embeddings.
3. **Detect bigrams (word pairs)** such as "blood sugar" and "high pressure", ensuring better representation.
4. **Analyze the trained embeddings** by finding words similar to "diabetes" and "hypertension".

In Natural Language Processing (NLP), a corpus (plural: corpora) refers to a large collection of text data that is used for training machine learning models, especially for language-based tasks like text classification, machine translation, and word embeddings.

Corpus in the Word2Vec Experiment

A corpus **in this context means** a set of sentences related to a specific domain **that is used to** train the Word2Vec model**. It provides** context **for words so that the model can learn their meanings and relationships**.

Prerequisites and Key Concepts

To understand this experiment fully, familiarity with the following concepts is beneficial:

1. **Word Representation in NLP**
   - One-hot encoding vs. dense embeddings
   - Need for vectorization of text data
2. **Word2Vec Model**
   - **CBOW (Continuous Bag of Words)**: Predicts a target word from its context.
   - **Skip-gram Model**: Predicts surrounding words for a given target word.
3. **Text Preprocessing Techniques**
   - Tokenization (splitting sentences into words)
   - Stopword removal (removing common words like "the", "is")

- Lemmatization (converting words to their root forms)

By the end of this experiment, students will develop an intuitive understanding of how word embeddings capture domain-specific relationships.

```
# Experiment 3: Word2Vec Training on Medical Corpus with Bigram Detection

# Import libraries

from gensim.models import Word2Vec

from gensim.models.phrases import Phrases, Phraser

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA



# Read medical corpus from an external file

with open("medical_corpus.txt", "r") as file:

    corpus = [line.strip() for line in file if line.strip()]



# Refined stopword list
```

```
stopwords = {"the", "a", "is", "for", "with", "to", "of", "and", "in", "can", "are"}

# Tokenization with stopword removal

tokenized_sentences = [

    [word for word in sentence.lower().split() if word not in stopwords]

    for sentence in corpus

]



# Detect bigrams to capture word combinations (e.g., "blood sugar", "high pressure")

bigram = Phrases(tokenized_sentences, min_count=2, threshold=5)

bigram_phraser = Phraser(bigram)

tokenized_sentences = [bigram_phraser[sentence] for sentence in tokenized_sentences]
```

```
# Train Word2Vec model with improved parameters

model = Word2Vec(tokenized_sentences, vector_size=150, window=5, min_count=2, epochs=300, sg=1, hs=1,
negative=0)

# Display similar words

diabetes_similar = [(word, round(sim, 2)) for word, sim in model.wv.most_similar("diabetes", topn=5)]

print("Words similar to 'diabetes':", diabetes_similar)

hypertension_similar = [(word, round(sim, 2)) for word, sim in model.wv.most_similar("hypertension",
topn=5)]

print("Words similar to 'hypertension':", hypertension_similar)
```

Output:

```
Words similar to 'diabetes': [('lifestyle', 0.12), ('disease.', 0.11), ('diet', 0.08), ('sugar', 0.07), ('diabetes.', 0.06)]

Words similar to 'hypertension': [('disease.', 0.09), ('risk', 0.05), ('high', 0.03), ('diet', 0.01), ('blood', 0.01)]
```

medical_corpus.txt

```
The patient was diagnosed with hypertension and diabetes.
Insulin therapy is recommended for patients with type 1 diabetes.
Symptoms of hypertension include headaches and dizziness.
Diabetes is characterized by high blood sugar levels.
Beta-blockers are commonly prescribed for hypertension.
Type 2 diabetes can be managed with diet and exercise.
High blood pressure is another term for hypertension.
Diabetes increases the risk of heart disease.
Hypertension is often linked to kidney disease.
A balanced diet can help manage both diabetes and hypertension.
Regular physical activity lowers the risk of hypertension.
Diabetes management includes blood sugar monitoring.
Uncontrolled hypertension can lead to stroke.
Lifestyle changes are essential for managing diabetes and hypertension.
Early diagnosis of diabetes helps in better management.
Excess salt intake contributes to hypertension.
Medication adherence is key for diabetes control.
Hypertension treatment includes lifestyle modifications and medications.
Obesity is a major risk factor for diabetes and hypertension.
Genetics plays a role in the development of hypertension and diabetes.
```

In this experiment, we successfully:

- Trained a Word2Vec model on a medical domain-specific corpus.
- Improved embeddings by detecting bigrams (e.g., "blood sugar", "high pressure").
- Found meaningful word relationships by identifying similar terms.

This approach effectively captures domain-specific semantics, making it useful for applications like medical NLP, clinical text analysis, and healthcare AI models.

Concepts Related to Lab Program 3:

1. Natural Language Processing (NLP) Basics

What is NLP?

Natural Language Processing (NLP) is a branch of artificial intelligence that enables computers to understand, interpret, and generate human language. It is widely used in text classification, machine translation, and chatbot development.

Why is NLP important in this experiment?

In this experiment, we use NLP techniques such as tokenization, stopword removal, and word embeddings to process text and extract meaningful insights from the medical corpus.

2. Word Embedding’s and Their Importance

What are word embedding's?

Word embeddings are dense numerical vector representations of words. They capture semantic meanings by positioning similar words closer together in a multi-dimensional space.

Why are word embedding's better than traditional representations?

Unlike one-hot encoding, which treats words independently, word embeddings capture contextual relationships. This helps in understanding domain-specific terminologies in medical texts.

How does Word2Vec create embedding's?

Word2Vec uses neural networks to learn vector representations by predicting words in a given context.

3. Word2Vec Algorithm

What is Word2Vec?

Word2Vec is a machine learning algorithm that learns word representations based on their contextual relationships. It produces word embeddings that can be used for various NLP tasks.

What are Skip-gram and CBOW models?

- **Skip-gram (sg=1)**: Predicts surrounding words given a target word. Works well with smaller datasets.
- **CBOW (sg=0)**: Predicts the target word based on its surrounding words. Works better for larger datasets.

Why do we use Skip-gram in this experiment?

Since we have a relatively small dataset, Skip-gram is chosen as it performs better when training on limited text data.

4. Tokenization and Stopword Removal

What is tokenization?

Tokenization is the process of breaking text into individual words or tokens. This allows the model to understand and process each word separately.

Why remove stopwords?

Stopwords are commonly used words such as "the", "is", and "for" that do not contribute much meaning to text analysis. Removing them ensures that embeddings focus on meaningful words in the corpus.

How is tokenization performed in this experiment?

Each sentence from the medical corpus is split into lowercase words, and stopwords are removed before training the Word2Vec model.

5. Bigram Detection (Phrase Modeling)

What are bigrams?

Bigrams are **pairs of words that frequently appear together**, such as "blood sugar" and "high pressure". Detecting bigrams ensures that the model treats them as single units rather than separate words.

How does bigram detection work in this experiment?

We use **Gensim’s Phrases and Phraser** to automatically detect and join frequent word pairs before training the Word2Vec model.

6. Similarity Measurement in Word2Vec

How does Word2Vec determine similar words?

Word2Vec finds similar words by calculating **cosine similarity** between word vectors. Words that appear in similar contexts have similar vector representations.

How is similarity measured in this experiment?

After training, we query the model for words similar to "diabetes" and "hypertension" using:

```
model.wv.most_similar("diabetes", topn=5)
```

This returns the **top 5 most similar words** to "diabetes" based on their learned embeddings.

7. Reading and Processing Data from External Files

Why do we load data from a file? Instead of manually defining a dataset, we store the **medical corpus in medical_corpus.txt**, making the experiment more scalable.

How is the file read in this experiment? The corpus is read using:

```
with open("medical_corpus.txt", "r") as file:
    corpus = [line.strip() for line in file if line.strip()]
```

This reads the file **line by line**, removes extra spaces, and stores it in a list for processing.