Lab 9.3: Working with Pre-trained Word Embeddings

2403A52006

MD.Mustafa

Import Required Libraries

```
!pip install gensim
import gensim.downloader as api
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
```

```
Collecting gensim
  Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (8.4 kB)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.12/dist-packages (from gensim) (2.0.2)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from gensim) (1.16.3)
Requirement already satisfied: smart_open>=1.8.1 in /usr/local/lib/python3.12/dist-packages (from gensim) (7.5.0
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart_open>=1.8.1->gensim)
Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl (27.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 27.9/27.9 MB 53.4 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.4.0
```

Load Pre-trained Word2Vec Model

```
print("Loading pre-trained Word2Vec model...")
model = api.load("word2vec-google-news-300")

print("\nModel Loaded Successfully!")
print("Vocabulary Size:", len(model.key_to_index))
```

```
Loading pre-trained Word2Vec model...
[==================================================] 100.0% 1662.8/1662.8MB downloaded

Model Loaded Successfully!
Vocabulary Size: 3000000
```

Display Example Word Vector

```
word = "king"
vector = model[word]

print(f"Vector for '{word}' (first 10 dimensions shown):\n")
print(vector[:10])
print("\nVector length:", len(vector))
```

```
Vector for 'king' (first 10 dimensions shown):

[ 0.12597656  0.02978516  0.00860596  0.13964844 -0.02563477 -0.03613281
  0.11181641 -0.19824219  0.05126953  0.36328125]

Vector length: 300
```

Word Similarity

```
word_pairs = [
    ("doctor", "nurse"),
    ("cat", "dog"),
    ("car", "bus"),
    ("king", "queen"),
    ("teacher", "student"),
    ("computer", "software"),
```

```
    ("computer", "software"),
    ("apple", "banana"),
    ("city", "village"),
    ("football", "cricket"),
    ("hospital", "medicine")
]

print("Word Similarity Results:\n")

for w1, w2 in word_pairs:
    similarity = model.similarity(w1, w2)
    print(f"{w1} ↔ {w2} : {similarity:.4f}")
```

```
Word Similarity Results:

doctor ↔ nurse : 0.6320
cat ↔ dog : 0.7609
car ↔ bus : 0.4693
king ↔ queen : 0.6511
teacher ↔ student : 0.6301
computer ↔ software : 0.5444
apple ↔ banana : 0.5318
city ↔ village : 0.4790
football ↔ cricket : 0.4597
hospital ↔ medicine : 0.3351
```

Nearest Neighbor Exploration

```
test_words = ["king", "university", "doctor", "computer", "india"]

for word in test_words:
    print(f"\nTop 5 words similar to '{word}':")
    similar_words = model.most_similar(word, topn=5)
    for sim_word, score in similar_words:
        print(f"  {sim_word} : {score:.4f}")
```

```
Top 5 words similar to 'king':
  kings : 0.7138
  queen : 0.6511
  monarch : 0.6413
  crown_prince : 0.6204
  prince : 0.6160

Top 5 words similar to 'university':
  universities : 0.7004
  faculty : 0.6781
  unversity : 0.6758
  undergraduate : 0.6587
  univeristy : 0.6585

Top 5 words similar to 'doctor':
  physician : 0.7806
  doctors : 0.7477
  gynecologist : 0.6948
  surgeon : 0.6793
  dentist : 0.6785

Top 5 words similar to 'computer':
  computers : 0.7979
  laptop : 0.6640
  laptop_computer : 0.6549
  Computer : 0.6473
  com_puter : 0.6082

Top 5 words similar to 'india':
  indian : 0.6967
  usa : 0.6836
  pakistan : 0.6815
  chennai : 0.6676
  america : 0.6589
```

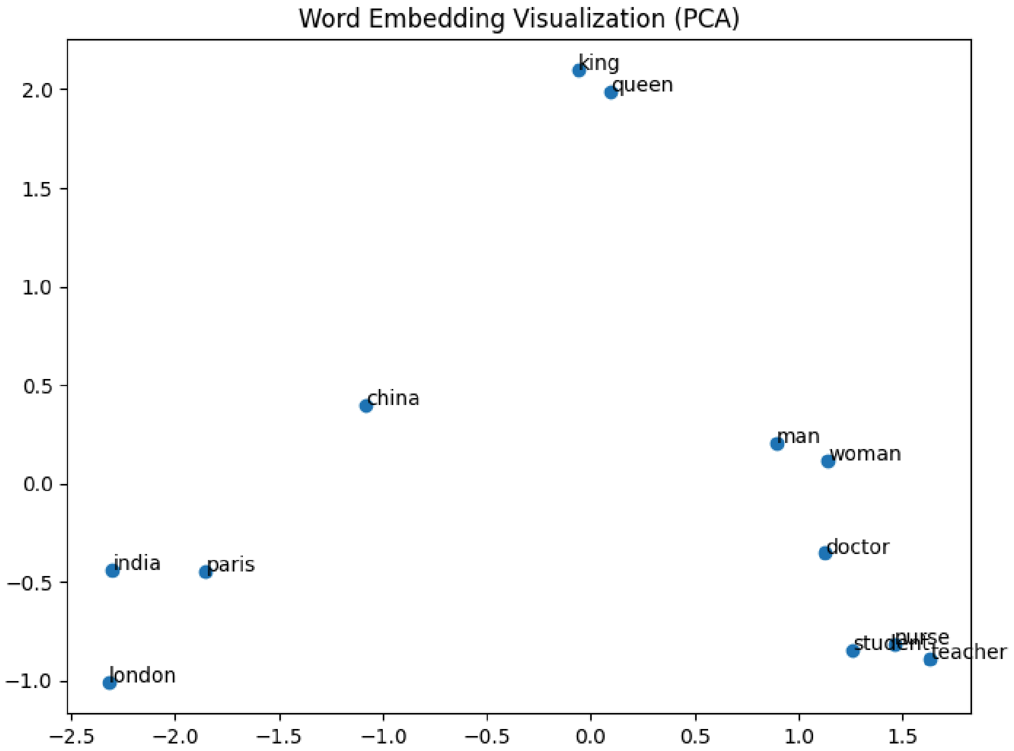
Word Analogy Tasks

```
analogies = [
    ("king", "man", "woman"),
    ("paris", "france", "india"),
    ("teacher", "school", "hospital"),
]

print("Analogy Results:\n")

for w1, w2, w3 in analogies:
    result = model.most_similar(positive=[w1, w3], negative=[w2], topn=1)
    print(f"{w1} - {w2} + {w3} ≈ {result[0][0]} ({result[0][1]:.4f})")
```

```
Analogy Results:

king - man + woman ≈ queen (0.7118)
paris - france + india ≈ chennai (0.5443)
teacher - school + hospital ≈ Hospital (0.6331)
```

Visualization using PCA

```
words = ["king", "queen", "man", "woman",
         "doctor", "nurse", "teacher", "student",
         "india", "china", "paris", "london"]

vectors = np.array([model[word] for word in words])

pca = PCA(n_components=2)
result = pca.fit_transform(vectors)

plt.figure(figsize=(8,6))
plt.scatter(result[:, 0], result[:, 1])

for i, word in enumerate(words):
    plt.annotate(word, xy=(result[i, 0], result[i, 1]))

plt.title("Word Embedding Visualization (PCA)")
plt.show()
```



Reflection & Interpretation

Word embeddings represent words as dense vectors that capture semantic meaning. Words appearing in similar contexts have similar vector representations. Similarity scores reflect semantic closeness rather than exact word matching.

Analogies work because embeddings encode relationships as vector differences. Clusters in visualization indicate thematic grouping (e.g., gender pairs, countries, professions). However, embeddings may reflect biases present in training data and may fail for rare words.