

```
!pip install gensim
```

```
import gensim.downloader as api
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.12/dist-packages (4.4.0)
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) (2.1.1)
```

```
# Load GloVe model
model = api.load("glove-wiki-gigaword-100")
```

```
print("Vocabulary Size:", len(model))
```

```
[=====] 100.0% 128.1/128.1MB downloaded
Vocabulary Size: 400000
```

```
print("Vector for 'king':")
print(model['king'])
```

```
Vector for 'king':
[-0.32307 -0.87616  0.21977  0.25268  0.22976  0.7388  -0.37954
 -0.35307 -0.84369 -1.1113  -0.30266  0.33178 -0.25113  0.30448
 -0.077491 -0.89815  0.092496 -1.1407  -0.58324  0.66869 -0.23122
 -0.95855  0.28262 -0.078848  0.75315  0.26584  0.3422  -0.33949
 0.95608  0.065641  0.45747  0.39835  0.57965  0.39267 -0.21851
 0.58795 -0.55999  0.63368 -0.043983 -0.68731 -0.37841  0.38026
 0.61641 -0.88269 -0.12346 -0.37928 -0.38318  0.23868  0.6685
 -0.43321 -0.11065  0.081723  1.1569  0.78958 -0.21223 -2.3211
 -0.67806  0.44561  0.65707  0.1045  0.46217  0.19912  0.25802
 0.057194  0.53443 -0.43133 -0.34311  0.59789 -0.58417  0.068995
 0.23944 -0.85181  0.30379 -0.34177 -0.25746 -0.031101 -0.16285
 0.45169 -0.91627  0.64521  0.73281 -0.22752  0.30226  0.044801
 -0.83741  0.55006 -0.52506 -1.7357  0.4751 -0.70487  0.056939
 -0.7132  0.089623  0.41394 -1.3363 -0.61915 -0.33089 -0.52881
 0.16483 -0.98878 ]
```

```
word_pairs = [
    ("doctor", "nurse"),
    ("cat", "dog"),
    ("car", "bus"),
    ("king", "queen"),
    ("apple", "orange"),
    ("teacher", "student"),
    ("river", "ocean"),
    ("man", "woman"),
    ("computer", "keyboard"),
    ("sun", "moon")
]
```

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

```
Similarity between doctor and nurse: 0.7522
Similarity between cat and dog: 0.8798
Similarity between car and bus: 0.7373
Similarity between king and queen: 0.7508
Similarity between apple and orange: 0.5007
Similarity between teacher and student: 0.8083
Similarity between river and ocean: 0.5743
Similarity between man and woman: 0.8323
Similarity between computer and keyboard: 0.5418
Similarity between sun and moon: 0.6138
```

```
words = ["king", "university", "money", "technology", "war"]
```

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

```

Top similar words for 'king':
[('prince', 0.7682328820228577), ('queen', 0.7507690787315369), ('son', 0.7020888328552246), ('brother', 0.6985775232315063), ('

Top similar words for 'university':
[('college', 0.8294212818145752), ('harvard', 0.8156033754348755), ('yale', 0.8113803267478943), ('professor', 0.810378491878509

Top similar words for 'money':
[('funds', 0.8508071303367615), ('cash', 0.848483681678772), ('fund', 0.7594833374023438), ('paying', 0.7415367364883423), ('pay

Top similar words for 'technology':
[('technologies', 0.8506267666816711), ('computer', 0.7642159461975098), ('tech', 0.7489413619041443), ('software', 0.7358859181

Top similar words for 'war':
[('wars', 0.7686851620674133), ('conflict', 0.7660517692565918), ('invasion', 0.7430229187011719), ('military', 0.73651087284088

```

```

print("king - man + woman =", model.most_similar(positive=["king", "woman"], negative=["man"], topn=1))

print("paris - france + india =", model.most_similar(positive=["paris", "india"], negative=["france"], topn=1))

print("teacher - school + hospital =", model.most_similar(positive=["teacher", "hospital"], negative=["school"], topn=1))

```

```

king - man + woman = [('queen', 0.7698540687561035)]
paris - france + india = [('delhi', 0.8654932975769043)]
teacher - school + hospital = [('nurse', 0.7798740267753601)]

```

```

words = ["king", "queen", "man", "woman", "doctor", "nurse",
         "paris", "france", "india", "delhi",
         "apple", "orange", "banana", "car", "bus"]

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

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

plt.figure(figsize=(10,6))
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()

```

