```
# Numerical computations
import numpy as np

# Install gensim if not already installed
!pip install gensim

# Word embedding handling
from gensim.models import KeyedVectors

# Visualization
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA

# Download files from internet
import requests
```

```
Collecting gensim
  Downloading gensim-4.4.0-cp312-cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl.metadata (8
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.12/dist-packages (from gens
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from gensi
Requirement already satisfied: smart_open>=1.8.1 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart_open>=
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 56.6 MB/s eta 0:00:00
Installing collected packages: gensim
Successfully installed gensim-4.4.0
```

```
url = "https://github.com/mmihaltz/word2vec-GoogleNews-vectors/raw/master/GoogleNews-vectors-negat
filename = "GoogleNews-vectors-negative300.bin.gz"

r = requests.get(url)

with open(filename, "wb") as f:
    f.write(r.content)

print("Model downloaded successfully")
```

```
Model downloaded successfully
```

```
model = KeyedVectors.load_word2vec_format(
    "GoogleNews-vectors-negative300.bin.gz",
    binary=True
)

print("Vocabulary size:", len(model.key_to_index))
print("Example vector for 'king':")
print(model["king"])
```

```
 -4.86373901e-04 -1.36718750e-01  3.24218750e-01 -2.46093750e-01
 -3.03649902e-03 -2.11914062e-01  1.25000000e-01  2.69531250e-01
  2.04101562e-01  8.25195312e-02 -2.01171875e-01 -1.60156250e-01
 -3.78417969e-02 -1.20117188e-01  1.15234375e-01 -4.10156250e-02
 -3.95507812e-02 -8.98437500e-02  6.34765625e-03  2.03125000e-01
  1.86523438e-01  2.73437500e-01  6.29882812e-02  1.41601562e-01
 -9.81445312e-02  1.38671875e-01  1.82617188e-01  1.73828125e-01
  1.73828125e-01 -2.37304688e-01  1.78710938e-01  6.34765625e-02
  2.36328125e-01 -2.08984375e-01  8.74023438e-02 -1.66015625e-01
 -7.91015625e-02  2.43164062e-01 -8.88671875e-02  1.26953125e-01
 -2.16796875e-01 -1.73828125e-01 -3.59375000e-01 -8.25195312e-02
 -6.49414062e-02  5.07812500e-02  1.35742188e-01 -7.47070312e-02
 -1.64062500e-01  1.15356445e-02  4.45312500e-01 -2.15820312e-01
 -1.11328125e-01 -1.92382812e-01  1.70898438e-01 -1.25000000e-01
  2.65502930e-03  1.92382812e-01 -1.74804688e-01  1.39648438e-01
  2.92968750e-01  1.13281250e-01  5.95703125e-02 -6.39648438e-02
  9.96093750e-02 -2.72216797e-02  1.96533203e-02  4.27246094e-02
 -2.46093750e-01  6.39648438e-02 -2.25585938e-01 -1.68945312e-01
  2.89916992e-03  8.20312500e-02  3.41796875e-01  4.32128906e-02
  1.32812500e-01  1.42578125e-01  7.61718750e-02  5.98144531e-02
 -1.19140625e-01  2.74658203e-03 -6.29882812e-02 -2.72216797e-02
 -4.82177734e-03 -8.20312500e-02 -2.49023438e-02 -4.00390625e-01
 -1.06933594e-01  4.24804688e-02  7.76367188e-02 -1.16699219e-01
  7.37304688e-02 -9.22851562e-02  1.07910156e-01  1.58203125e-01
  4.24804688e-02  1.26953125e-01  3.61328125e-02  2.67578125e-01
 -1.01074219e-01 -3.02734375e-01 -5.76171875e-02  5.05371094e-02
  5.26428223e-04 -2.07031250e-01 -1.38671875e-01 -8.97216797e-03
 -2.78320312e-02 -1.41601562e-01  2.07031250e-01 -1.58203125e-01
  1.27929688e-01  1.49414062e-01 -2.24609375e-02 -8.44726562e-02
  1.22558594e-01  2.15820312e-01 -2.13867188e-01 -3.12500000e-01
 -3.73046875e-01  4.08935547e-03  1.07421875e-01  1.06933594e-01
  7.32421875e-02  8.97216797e-03 -3.88183594e-02 -1.29882812e-01
  1.49414062e-01 -2.14843750e-01 -1.83868408e-03  9.91210938e-02
  1.57226562e-01 -1.14257812e-01 -2.05078125e-01  9.91210938e-02
  3.69140625e-01 -1.97265625e-01  3.54003906e-02  1.09375000e-01
  1.31835938e-01  1.66992188e-01  2.35351562e-01  1.04980469e-01
 -4.96093750e-01 -1.64062500e-01 -1.56250000e-01 -5.22460938e-02
  1.03027344e-01  2.43164062e-01 -1.88476562e-01  5.07812500e-02
 -9.37500000e-02 -6.68945312e-02  2.27050781e-02  7.61718750e-02
  2.89062500e-01  3.10546875e-01 -5.37109375e-02  2.28515625e-01
  2.51464844e-02  6.78710938e-02 -1.21093750e-01 -2.15820312e-01
 -2.73437500e-01 -3.07617188e-02 -3.37890625e-01  1.53320312e-01
  2.33398438e-01 -2.08007812e-01  3.73046875e-01  8.20312500e-02
  2.51953125e-01 -7.61718750e-02 -4.66308594e-02 -2.23388672e-02
  2.99072266e-02 -5.93261719e-02 -4.66918945e-03 -2.44140625e-01
 -2.09960938e-01 -2.87109375e-01 -4.54101562e-02 -1.77734375e-01
 -2.79296875e-01 -8.59375000e-02  9.13085938e-02  2.51953125e-01]
```

```python
pairs = [
    ("doctor", "nurse"),
    ("cat", "dog"),
    ("car", "bus"),
    ("king", "queen"),
    ("apple", "orange"),
    ("teacher", "student"),
    ("computer", "laptop"),
    ("city", "village"),
    ("man", "woman"),
    ("sun", "moon")
]

# Calculate and print the similarity between each pair of words
for w1, w2 in pairs:
    print(f"{w1} - {w2}: {model.similarity(w1, w2)}")
```

```
doctor - nurse: 0.6319523453712463
cat - dog: 0.760945737361908
car - bus: 0.4693371057510376
king - queen: 0.6510956883430481
apple - orange: 0.3920346200466156
teacher - student: 0.6301365494728088
computer - laptop: 0.6640492677688599
city - village: 0.47896867990493774
man - woman: 0.7664012312889099
sun - moon: 0.4262833893299103
```

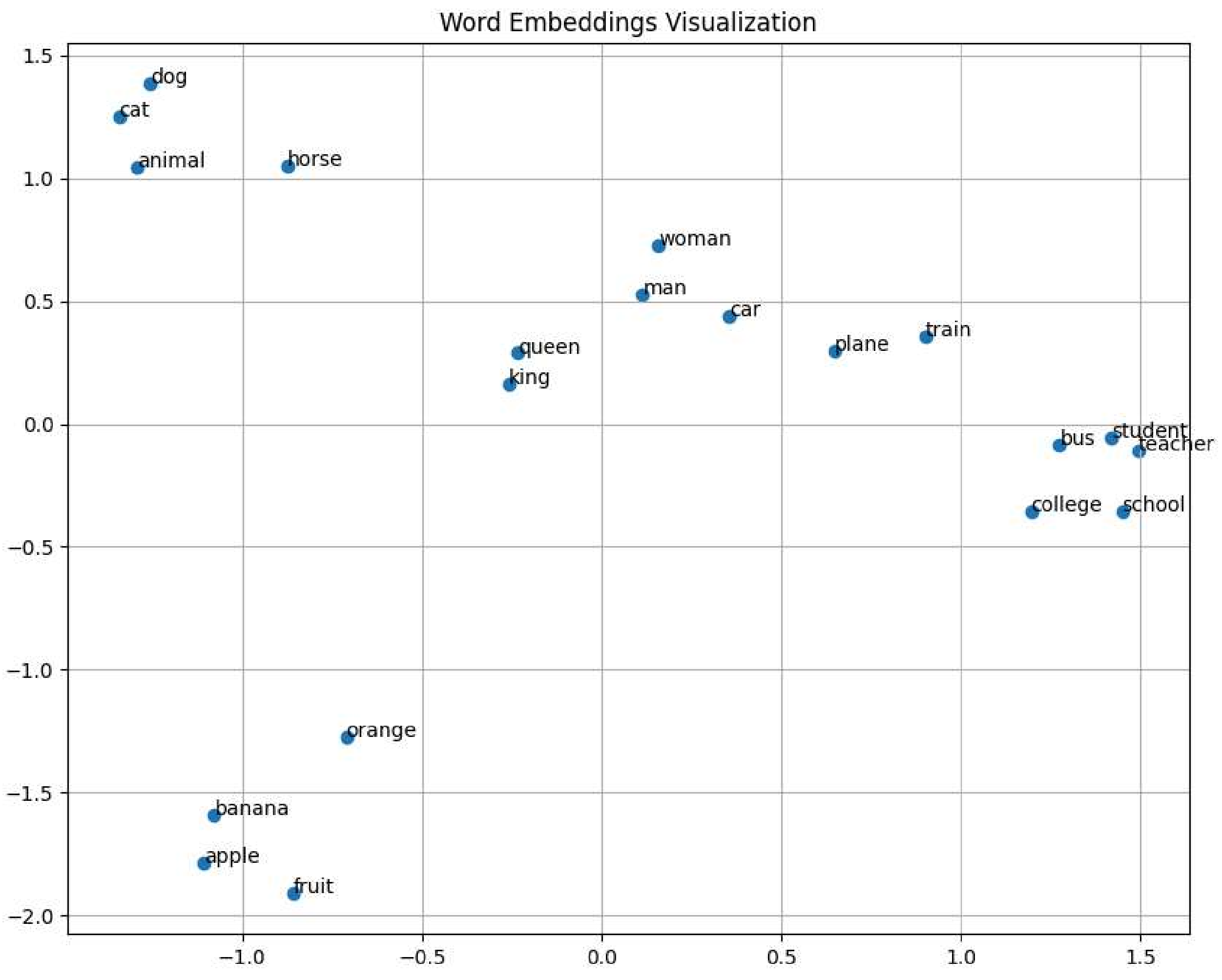
```
words = ["king", "university", "car", "computer", "music"]

# Find and print the top 5 most similar words for each given word
for word in words:
    print(f"\nSimilar words for {word}")
    print(model.most_similar(word, topn=5))
```

```

Similar words for king
[('kings', 0.7138045430183411), ('queen', 0.6510956883430481), ('monarch', 0.6413194537162781), ('

Similar words for university
[('universities', 0.7003918886184692), ('faculty', 0.6780907511711121), ('unversity', 0.6758289933

Similar words for car
[('vehicle', 0.7821096181869507), ('cars', 0.7423831224441528), ('SUV', 0.7160962224006653), ('min

Similar words for computer
[('computers', 0.7979379892349243), ('laptop', 0.6640493273735046), ('laptop_computer', 0.65488684

Similar words for music
[('classical_music', 0.7197794318199158), ('jazz', 0.6834640502929688), ('Music', 0.65957206487655
```

```
# Perform word analogy tasks (e.g., king - man + woman = queen)

# Analogy 1: king - man + woman
print(model.most_similar(
    positive=['king','woman'],
    negative=['man'],
    topn=1))

# Analogy 2: paris - france + india
print(model.most_similar(
    positive=['paris','india'],
    negative=['france'],
    topn=1))

# Analogy 3: teacher - school + hospital
print(model.most_similar(
    positive=['teacher','hospital'],
    negative=['school'],
    topn=1))
```

```
[('queen', 0.7118193507194519)]
[('chennai', 0.5442505478858948)]
[('Hospital', 0.6331106424331665)]
```

```
# Define a list of words for visualization
words = ["king","queen","man","woman",
         "car","bus","train","plane",
         "apple","banana","orange","fruit",
         "dog","cat","horse","animal",
         "teacher","student","school","college"]

# Get the vector representation for each word
vectors = np.array([model[w] for w in words])

# Apply Principal Component Analysis (PCA) to reduce dimensions to 2 for visualization
pca = PCA(n_components=2)
reduced = pca.fit_transform(vectors)

# Create a scatter plot of the reduced word vectors
plt.figure(figsize=(10, 8))
plt.scatter(reduced[:,0], reduced[:,1])

# Annotate each point with its corresponding word
for i, word in enumerate(words):
    plt.annotate(word, (reduced[i,0], reduced[i,1]))

# Set plot title and display the plot
plt.title("Word Embeddings Visualization")
plt.grid(True)
plt.show()
```


Word Embeddings Visualization
dog
cat
animal
horse
woman
man
car
train
plane
queen
king
bus
student
teacher
college
school
orange
banana
apple
fruit