```
!pip install gensim
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.12/dist-packages (4.3.3)
Requirement already satisfied: numpy<2.0,>=1.18.5 in /usr/local/lib/python3.12/dist-packages (from gensim) (1.26.4)
Requirement already satisfied: scipy<1.14.0,>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from gensim) (1.13.1)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.12/dist-packages (from gensim) (7.3.1)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart-open>=1.8.1->gensim) (1.17.3)
```

```
import gensim
from gensim.models import KeyedVectors
import gensim.downloader as api
import numpy as np
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
from sklearn.metrics.pairwise import cosine_similarity
```

```
# Tải mô hình GloVe đã huấn luyện sẵn
print("Đang tải mô hình GloVe...")
model = api.load("glove-wiki-gigaword-100")  # Mô hình GloVe 100 chiều
print("Mô hình đã tải xong!")
```

```
Đang tải mô hình GloVe...
Mô hình đã tải xong!
```

```
# Hiển thị số từ và số chiều
model.vectors.shape
```

```
(400000, 100)
```

```
# Lấy danh sách từ và vector
words = list(model.key_to_index.keys())
word_vectors = np.array([model[word] for word in words])
```

```
# Tính toán độ tương đồng cosine
def find_similar_words(word, model, word_vectors, words, top_k=5):
    if word not in model.key_to_index:
        print(f"Từ '{word}' không có trong từ điển.")
        return
    # Tính độ tương đồng cosine
    similarities = cosine_similarity([model[word]], word_vectors)[0]
    # Lấy chỉ số của Top K từ tương đồng
    top_k_indices = similarities.argsort()[-top_k-1:-1][::-1]
    print(f" Top {top_k} từ tương đồng với '{word}':")
    for idx in top_k_indices:
        print(f"- {words[idx]}: {similarities[idx]:.4f}")
```

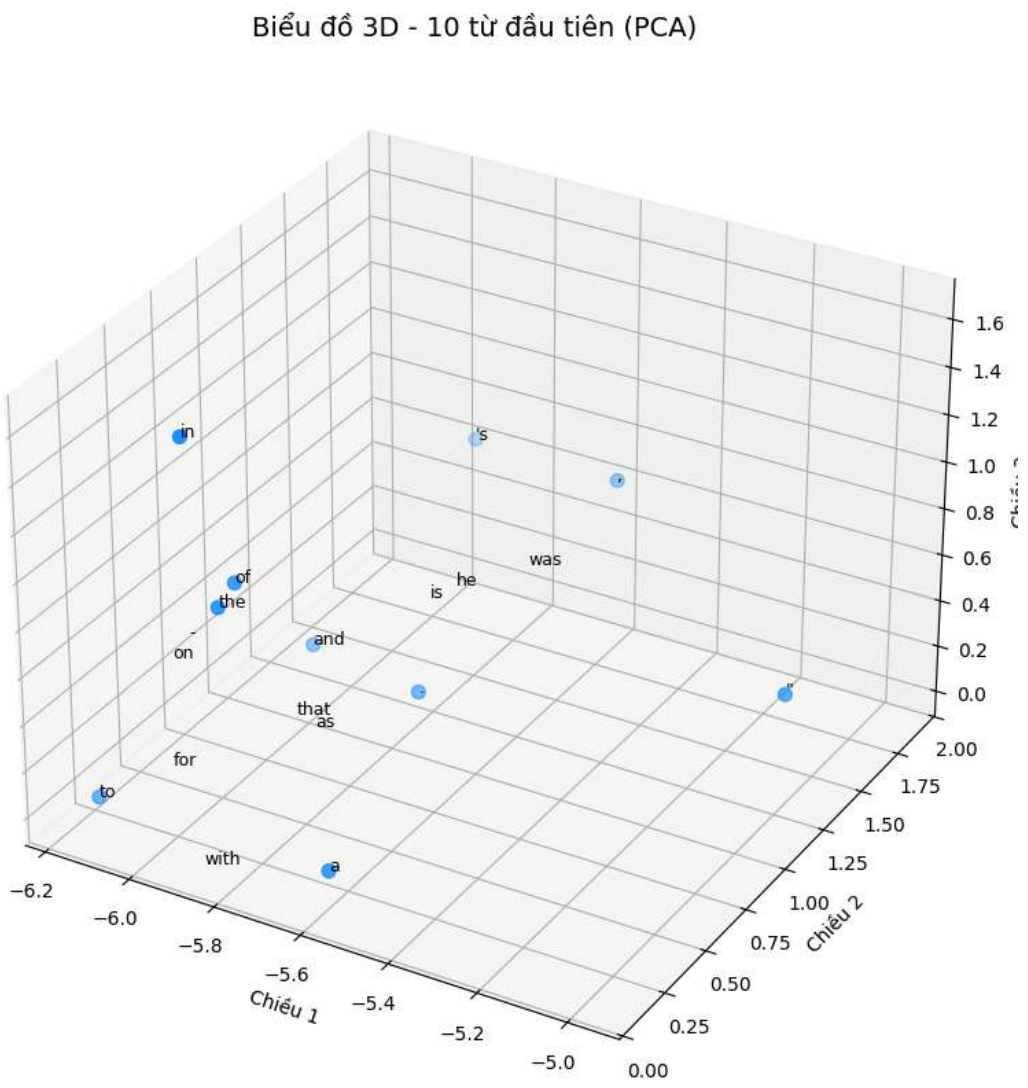
```
# Tìm kiếm với một từ cụ thể
find_similar_words("woman", model, word_vectors, words, top_k=5)
```

```
 Top 5 từ tương đồng với 'woman':
- girl: 0.8473
- man: 0.8323
- mother: 0.8276
- boy: 0.7721
- she: 0.7632
```

```
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D
import random

# Giảm chiều từ 100D → 3D bằng PCA
pca = PCA(n_components=3, random_state=42)
word_vectors_3d = pca.fit_transform(word_vectors)
```

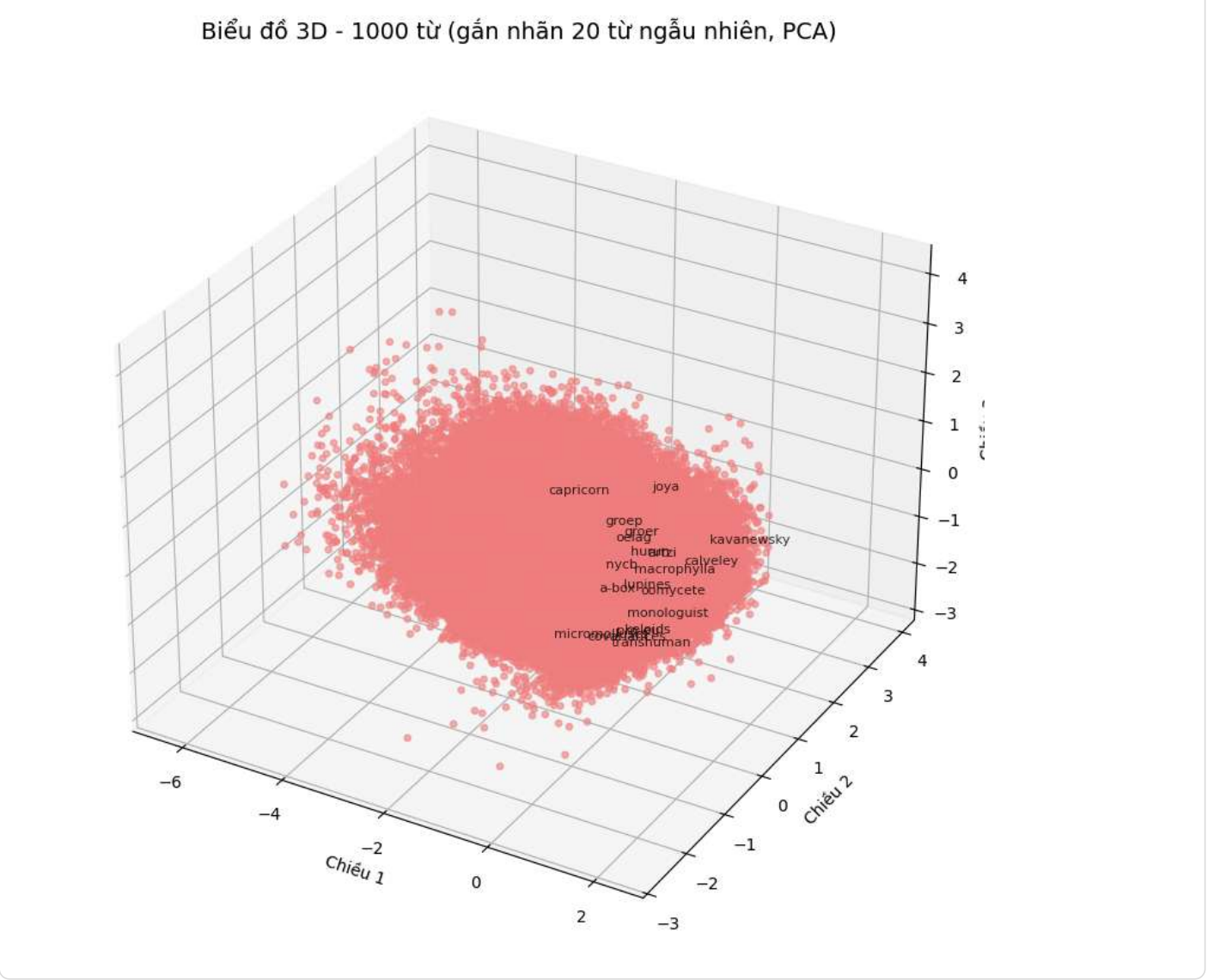
```
# Biểu đồ 1 – hiển thị 20 từ đầu tiên
fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111, projection='3d')
```

```
ax.scatter(word_vectors_3d[:10, 0], word_vectors_3d[:10, 1], word_vectors_3d[:10, 2],
           color='dodgerblue', s=50)

# Gắn nhãn cho 20 từ đầu tiên
for i, word in enumerate(words[:20]):
    ax.text(word_vectors_3d[i, 0], word_vectors_3d[i, 1], word_vectors_3d[i, 2],
            word, fontsize=9, color='black')

ax.set_title("Biểu đồ 3D - 10 từ đầu tiên (PCA)", fontsize=14)
ax.set_xlabel("Chiều 1")
ax.set_ylabel("Chiều 2")
ax.set_zlabel("Chiều 3")
plt.show()
```




```
# Biểu đồ 2 – hiển thị toàn bộ 40000 từ (gắn nhãn 20 từ ngẫu nhiên)
fig = plt.figure(figsize=(12, 10))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(word_vectors_3d[:, 0], word_vectors_3d[:, 1], word_vectors_3d[:, 2],
           color='lightcoral', s=15, alpha=0.6)

# Gắn nhãn cho 20 từ ngẫu nhiên
label_indices = random.sample(range(len(words)), 20)
for i in label_indices:
    ax.text(word_vectors_3d[i, 0], word_vectors_3d[i, 1], word_vectors_3d[i, 2],
            words[i], fontsize=8, color='black', alpha=0.8)

ax.set_title("Biểu đồ 3D - 1000 từ (gắn nhãn 20 từ ngẫu nhiên, PCA)", fontsize=14)
ax.set_xlabel("Chiều 1")
ax.set_ylabel("Chiều 2")
```

```
ax.set_zlabel("Chiều 3")
plt.show()
```



- 1. Nhận xét về phương pháp triển khai (PCA)

PCA (Principal Component Analysis) là một phương pháp giảm chiều tuyến tính, hoạt động bằng cách tìm ra các trục (thành phần chính) mà dữ liệu biến thiên mạnh nhất.

So với t-SNE hay UMAP, PCA đơn giản và nhanh hơn rất nhiều, nên phù hợp khi cần trực quan hóa nhanh hoặc tập dữ liệu lớn (như 1000+ từ).

PCA giữ được cấu trúc toàn cục (global structure) tốt — nghĩa là các nhóm từ cách xa nhau trong không gian 100D vẫn xa nhau trong không gian 3D.

Tuy nhiên, PCA không thể hiện tốt cấu trúc cục bộ (local clusters) — nên các từ đồng nghĩa gần nhau về ngữ nghĩa có thể chưa cụm lại rõ ràng như khi dùng t-SNE.

- 2. Nhận xét về kết quả quan sát được

Khi quan sát biểu đồ 3D với 20 từ đầu tiên, các điểm (vector từ) được phân bố rải rác trong không gian, cho thấy mỗi từ có hướng (vector) khác nhau trong embedding space.

Cấu trúc tổng thể trong PCA 3D vẫn phản ánh độ tương đồng ngữ nghĩa ở mức độ khái quát, giúp kiểm tra nhanh chất lượng embedding