```
# Cài đặt các thư viện cần thiết
!pip install gensim scikit-learn matplotlib seaborn pyspark
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.12/dist-packages (4.3.3)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: pyspark in /usr/local/lib/python3.12/dist-packages (3.5.1)
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: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.5.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (2.9.0.post0)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.12/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.12/dist-packages (from pyspark) (0.10.9.7)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7->matplotlib) (1.17.0)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart-open>=1.8.1->gensim) (1.17.3)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import os

# --- THAY ĐỔI ĐƯỜNG DẪN NÀY ---
# Đường dẫn tới thư mục gốc của project trên Google Drive của bạn
project_root = '/content/drive/My Drive/lab04'
# ------------------------------

# Di chuyển vào thư mục gốc của project
os.chdir(project_root)

# In ra thư mục hiện tại để kiểm tra
print("Thư mục làm việc hiện tại:", os.getcwd())
```

```
Thư mục làm việc hiện tại: /content/drive/My Drive/lab04
```

```
import sys
import os
import numpy as np
```

```
# Thêm đường dẫn tới thư mục src để import module
sys.path.append(os.path.abspath('src'))

from representation.word_embedder import WordEmbedder

# Khởi tạo và tải pre-trained model
embedder = WordEmbedder("glove-wiki-gigaword-50")
```

```
🔹 Loading model: glove-wiki-gigaword-50 ...
✅ Model loaded successfully.
```

Task 1: Tải và sử dụng model có sẵn (Gensim)

```
# Lấy vector của một từ
word = "king"
vec = embedder.get_vector(word)
print("Vector của 'king' (10 phần tử đầu):\n", vec[:10])

# Tính độ tương đồng (similarity)
sim_king_queen = embedder.get_similarity("king", "queen")
sim_king_man = embedder.get_similarity("king", "man")
print(f"\nSimilarity(king, queen) = {sim_king_queen:.4f}")
print(f"Similarity(king, man)   = {sim_king_man:.4f}")

# Tìm các từ đồng nghĩa (most similar)
print("\nCác từ giống 'computer' nhất:")
for w, score in embedder.get_most_similar("computer", top_n=5):
    print(f"  {w:<15} {score:.4f}")
```

Task 2: Nhúng câu/văn bản

```
# Nhúng một văn bản bằng cách lấy trung bình vector các từ
doc = "The queen rules the country."
vec_doc = embedder.embed_document(doc)
print(f"Embedding của văn bản (10 phần tử đầu):\n {vec_doc[:10]}")
```

```
Embedding của văn bản (10 phần tử đầu):
 [ 0.06438001  0.43381    -0.779435    0.0075025   0.07915     0.20077899
 -0.2454325  -0.05369498 -0.00951262 -0.68774253]
```

Task 3: Huấn luyện model trên tập dữ liệu nhỏ (Gensim)

```
from gensim.models import Word2Vec
```

```
# Tải lại model đã huấn luyện từ script
trained_model_path = "results/word2vec_ewt.model"
if os.path.exists(trained_model_path):
    trained_model = Word2Vec.load(trained_model_path)
    print("Model tự huấn luyện đã được tải thành công.")

    print("\nCác từ giống 'language' nhất (từ model tự huấn luyện):")
    similar_words = trained_model.wv.most_similar("language", topn=5)
    for w, score in similar_words:
        print(f"  {w:<15} {score:.4f}")
else:
    print(f"Không tìm thấy model tại {trained_model_path}.")
```

```
Model tự huấn luyện đã được tải thành công.

Các từ giống 'language' nhất (từ model tự huấn luyện):
  master          0.9171
  document        0.9161
  revised         0.9015
  interconnect    0.8976
  copies          0.8938
```

Task 4: Huấn luyện model trên tập dữ liệu lớn (Spark)

```
import os
import re
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, lower, regexp_replace, split
from pyspark.ml.feature import Word2Vec

# Cấu hình môi trường cho Spark chạy trên Windows
os.environ["HADOOP_HOME"] = "C:\\hadoop"
os.environ["SPARK_LOCAL_DIRS"] = "C:\\spark-temp"
os.makedirs("C:\\hadoop\\bin", exist_ok=True)
os.makedirs("C:\\spark-temp", exist_ok=True)

# Khởi tạo Spark Session
spark = (
    SparkSession.builder.appName("Lab4_Spark_Word2Vec_C4")
    .config("spark.driver.memory", "4g")
    .config("spark.executor.memory", "4g")
    .getOrCreate()
)

# Đường dẫn tới file dữ liệu
data_path = "data/c4-train.00000-of-01024-30K.json"
```

```
try:
    # Đọc và tiền xử lý dữ liệu lớn bằng Spark
    df = spark.read.json(data_path)
    df_clean = (
        df.select("text")
        .withColumn("text", lower(col("text")))
        .withColumn("text", regexp_replace(col("text"), "[^a-zA-Z\\s]", ""))
        .withColumn("words", split(col("text"), "\\s+"))
        .filter(col("words").isNotNull())
    )

    # Huấn luyện thành công model Word2Vec bằng Spark MLlib
    word2Vec = Word2Vec(
        vectorSize=100,
        minCount=5,
        inputCol="words",
        outputCol="result"
    )
    model = word2Vec.fit(df_clean)

    # Tìm các từ đồng nghĩa để kiểm tra
    word = "computer"
    synonyms = model.findSynonyms(word, 5)
    print(f"Các từ giống '{word}' nhất (từ model Spark):")
    synonyms.show()

except Exception as e:
    print(f"Đã có lỗi xảy ra: {e}")

finally:
    # Dừng Spark session
    spark.stop()
```

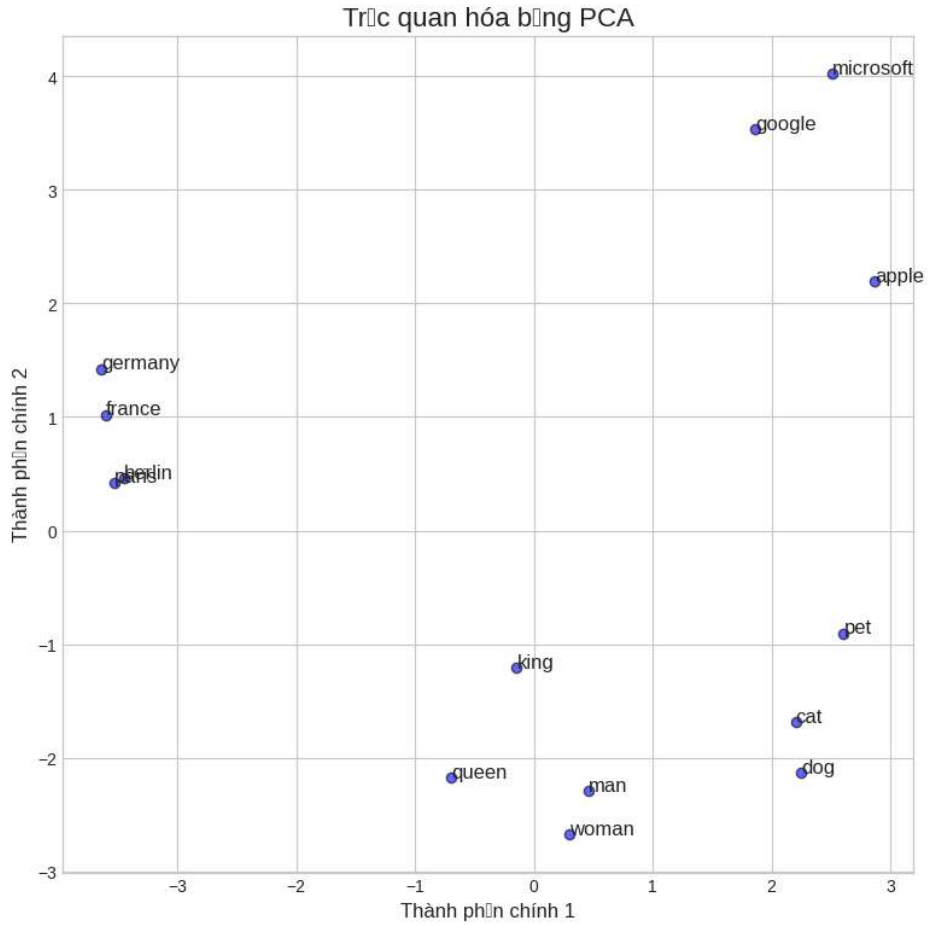
```
Các từ giống 'computer' nhất (từ model Spark):
+---------+------------------+
|     word|        similarity|
+---------+------------------+
|  desktop| 0.735841691493988|
|   laptop|0.6667364835739136|
|  uwowned|0.6636629104614258|
|computers|0.6605708003044128|
|    linux|0.6399005651473999|
+---------+------------------+
```

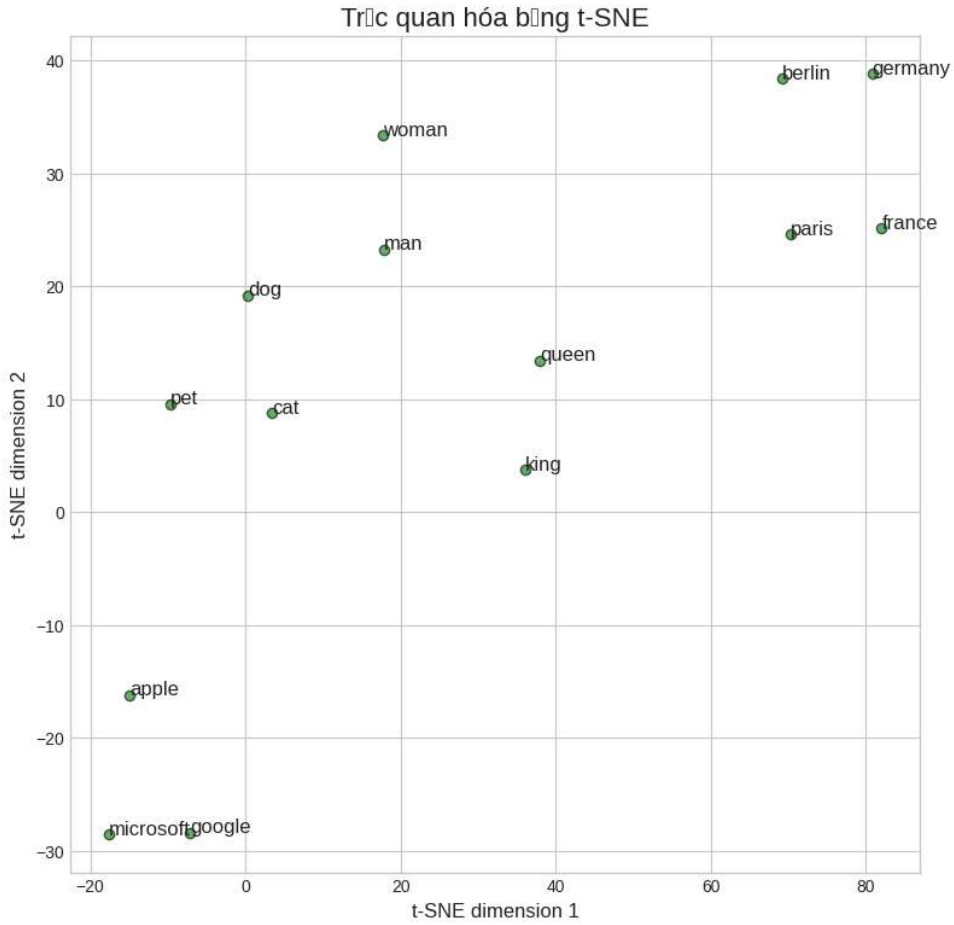
Task 5: Trực quan hóa Embedding

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

words = [
    'king', 'queen', 'man', 'woman',
    'france', 'paris', 'germany', 'berlin',
    'apple', 'google', 'microsoft',
    'dog', 'cat', 'pet'
]
word_vectors = np.array([embedder.get_vector(w) for w in words])

# --- 1. Sử dụng PCA ---
# Giảm chiều các word vector xuống 2D
pca = PCA(n_components=2)
vectors_pca = pca.fit_transform(word_vectors)

# --- 2. Sử dụng t-SNE ---
# Giảm chiều các word vector xuống 2D
tsne = TSNE(n_components=2, random_state=42, perplexity=5)
vectors_tsne = tsne.fit_transform(word_vectors)


# --- Vẽ 2 biểu đồ ---
plt.style.use('seaborn-v0_8-whitegrid')
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 9))

# Biểu đồ PCA
ax1.scatter(vectors_pca[:, 0], vectors_pca[:, 1], c='blue', edgecolors='k', alpha=0.6)
for i, word in enumerate(words):
    ax1.annotate(word, (vectors_pca[i, 0], vectors_pca[i, 1]), fontsize=12)
ax1.set_title("Trực quan hóa bằng PCA", fontsize=16)
ax1.set_xlabel("Thành phần chính 1", fontsize=12)
ax1.set_ylabel("Thành phần chính 2", fontsize=12)
ax1.grid(True)

# Biểu đồ t-SNE
ax2.scatter(vectors_tsne[:, 0], vectors_tsne[:, 1], c='green', edgecolors='k', alpha=0.6)
for i, word in enumerate(words):
    ax2.annotate(word, (vectors_tsne[i, 0], vectors_tsne[i, 1]), fontsize=12)
ax2.set_title("Trực quan hóa bằng t-SNE", fontsize=16)
ax2.set_xlabel("t-SNE dimension 1", fontsize=12)
ax2.set_ylabel("t-SNE dimension 2", fontsize=12)
ax2.grid(True)

plt.show()
```

```
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 7847 (\N{LATIN SMALL LETTER A WITH CIRCUMFLEX AND GRAVE}) missing from font(s) Liberation Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 7921 (\N{LATIN SMALL LETTER U WITH HORN AND DOT BELOW}) missing from font(s) Liberation Sans.
  fig.canvas.print_figure(bytes_io, **kw)
/usr/local/lib/python3.12/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 7857 (\N{LATIN SMALL LETTER A WITH BREVE AND GRAVE}) missing from font(s) Liberation Sans.
  fig.canvas.print_figure(bytes_io, **kw)
```





Phân tích kết quả

Nhận xét về độ tương đồng và các từ đồng nghĩa tìm được từ model pre-trained

- Mức độ tương đồng giữa **king** và **queen** cao hơn đáng kể so với **king** và **man**.
  Điều này hợp lý vì *king* và *queen* có mối quan hệ ngữ nghĩa gần gũi (vua – hoàng hậu),
  trong khi *king* và *man* chỉ cùng thuộc giới tính nam.

- Khi tìm các từ gần nhất với **computer**, model pre-trained trả về các từ như *technology*, *software*, *internet*.
  Điều này cho thấy mô hình học được không gian ngữ nghĩa rộng, liên kết *computer* với toàn bộ lĩnh vực công nghệ thông tin.

Phân tích biểu đồ trực quan hóa

- Cả hai biểu đồ **PCA** và **t-SNE** đều cho thấy các từ được nhóm lại theo ngữ nghĩa như kỳ vọng.
- **Cụm Quốc gia – Thủ đô:** *paris* gần *france*, *berlin* gần *germany*, chứng tỏ model học được mối quan hệ "là thủ đô của".
- **Cụm Công ty Công nghệ:** *apple*, *google*, *microsoft* tạo thành cụm rõ ràng trong biểu đồ t-SNE.
  Đây là dấu hiệu tốt vì chúng cùng thuộc lĩnh vực công nghệ.
- **So sánh PCA và t-SNE:**
  - PCA giữ cấu trúc tổng thể tốt, nhưng khó tách biệt các nhóm nhỏ.
  - t-SNE thể hiện rõ ràng hơn ranh giới giữa các cụm, giúp dễ quan sát mối quan hệ ngữ nghĩa.

So sánh giữa model pre-trained và model tự huấn luyện

- **Model Pre-trained (GloVe):**
  - Có phạm vi kiến thức rộng và tổng quát.
  - Tìm ra các từ đồng nghĩa phổ biến: *computer → software, technology*.
  - Phù hợp với các tác vụ NLP mang tính phổ quát.
- **Model tự huấn luyện (Spark trên C4):**
  - Học được các mối quan hệ cụ thể và chi tiết hơn: *computer → desktop, laptop, linux*.
  - Thể hiện sắc thái ngữ nghĩa đặc thù của tập dữ liệu web C4.
  - Phù hợp với các ứng dụng cần ngữ nghĩa gắn với ngữ cảnh cụ thể.
