```
!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)
```

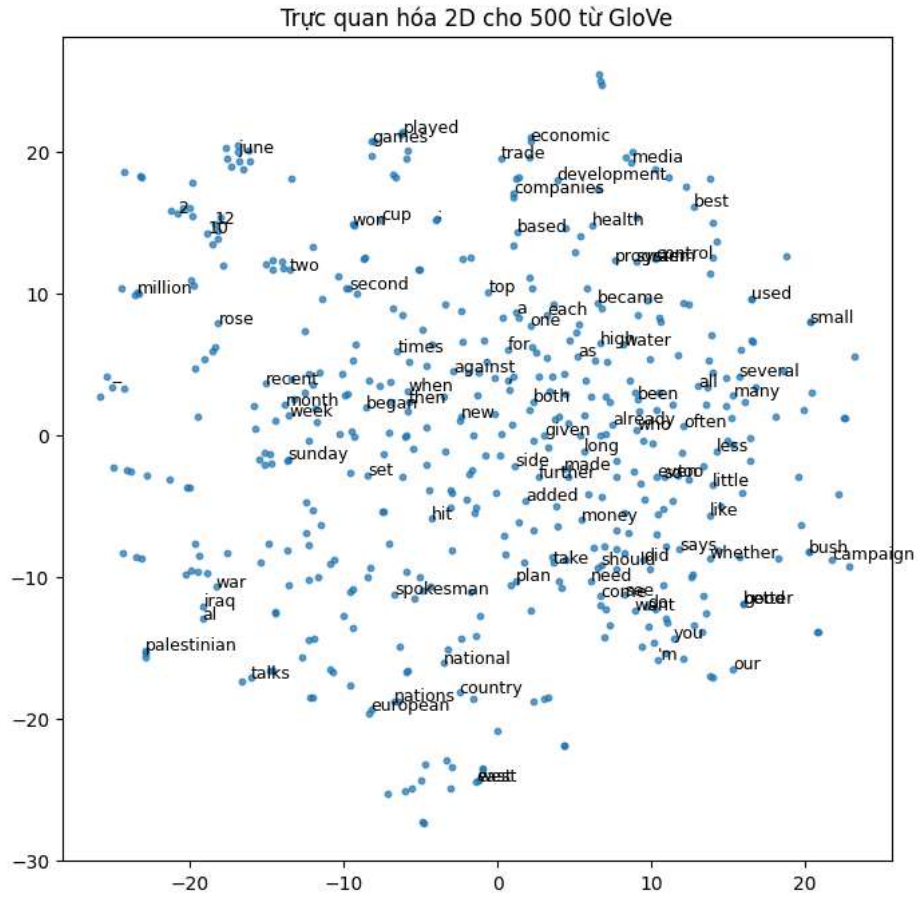
```
import gensim.downloader as api
import numpy as np
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import plotly.express as px
import random
```

GIẢM CHIỀU VÀ TRỰC QUAN HÓA

GLOVE

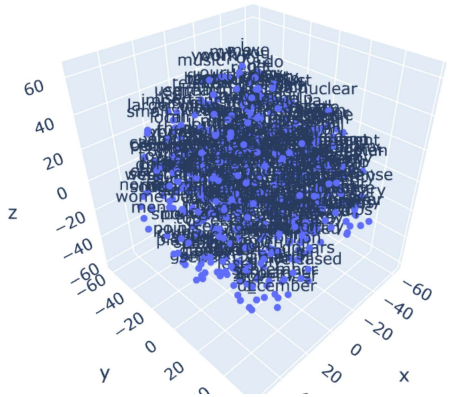
```
glove_model = api.load('glove-wiki-gigaword-100')

N_glove = 500
glove_vocab = list(glove_model.index_to_key)[:N_glove]
glove_vectors = glove_model[glove_vocab]

tsne_glove_2d = TSNE(n_components=2, random_state=42, perplexity=30)
glove_vectors_2d = tsne_glove_2d.fit_transform(glove_vectors)

tsne_glove_3d = TSNE(n_components=3, random_state=42, perplexity=30)
glove_vectors_3d = tsne_glove_3d.fit_transform(glove_vectors)
```

```
# Trực quan hóa 2D
plt.figure(figsize=(8, 8))
plt.scatter(glove_vectors_2d[:, 0], glove_vectors_2d[:, 1], s=10, alpha=0.7)
indices_glove = np.random.choice(range(len(glove_vocab)), 100, replace=False)
for i in indices_glove:
    plt.text(glove_vectors_2d[i, 0], glove_vectors_2d[i, 1], glove_vocab[i], fontsize=9)
plt.title(f'Trực quan hóa 2D cho {N_glove} từ GloVe')
plt.show()
```




```
# Trực quan hóa 3D
fig_glove = px.scatter_3d(
    x=glove_vectors_3d[:, 0],
    y=glove_vectors_3d[:, 1],
    z=glove_vectors_3d[:, 2],
    text=glove_vocab,
    title=f'Trực quan hóa 3D tương tác cho {N_glove} từ GloVe'
)
fig_glove.update_traces(marker=dict(size=3))
fig_glove.update_layout(width=800, height=500)
fig_glove.show()
```

Trực quan hóa 3D tương tác cho 500 từ GloVe

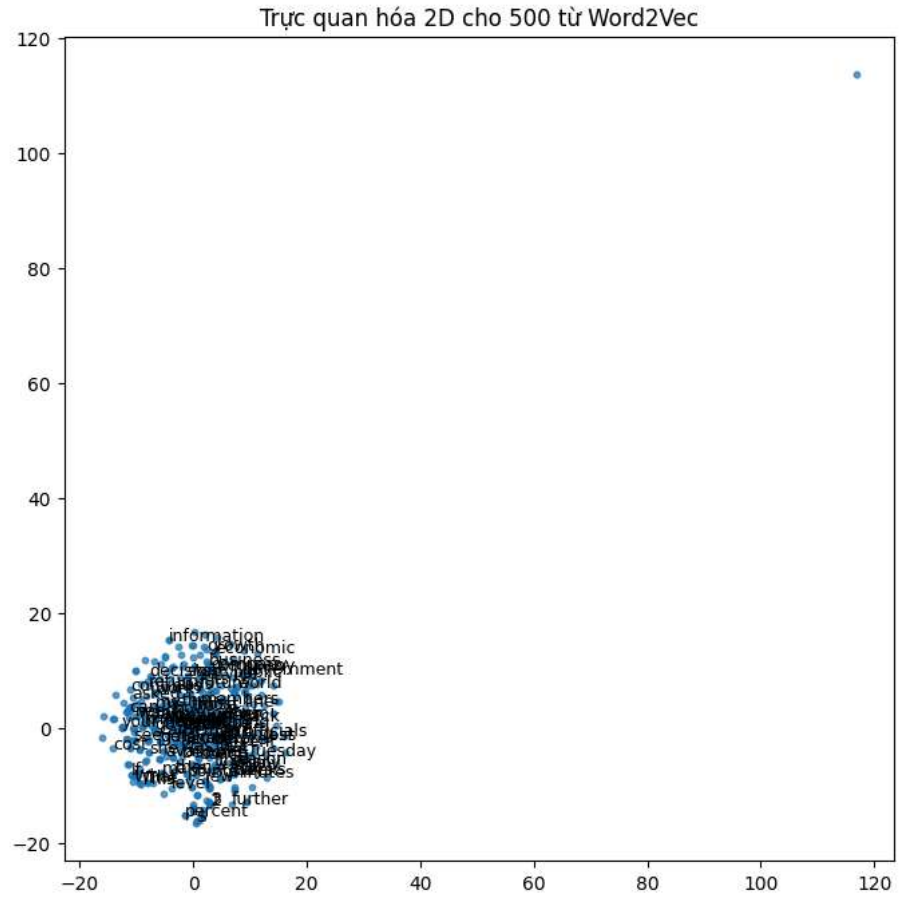


WORD2VEC

```
word2vec_model = api.load('word2vec-google-news-300')

N_word2vec = 500
word2vec_vocab = list(word2vec_model.index_to_key)[:N_word2vec]
word2vec_vectors = word2vec_model[word2vec_vocab]

tsne_word2vec_2d = TSNE(n_components=2, random_state=42, perplexity=30, n_iter=300)
word2vec_vectors_2d = tsne_word2vec_2d.fit_transform(word2vec_vectors)

tsne_word2vec_3d = TSNE(n_components=3, random_state=42, perplexity=30, n_iter=300)
word2vec_vectors_3d = tsne_word2vec_3d.fit_transform(word2vec_vectors)
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/manifold/_t_sne.py:1164: FutureWarning:

'n_iter' was renamed to 'max_iter' in version 1.5 and will be removed in 1.7.

/usr/local/lib/python3.12/dist-packages/sklearn/manifold/_t_sne.py:1164: FutureWarning:

'n_iter' was renamed to 'max_iter' in version 1.5 and will be removed in 1.7.
```

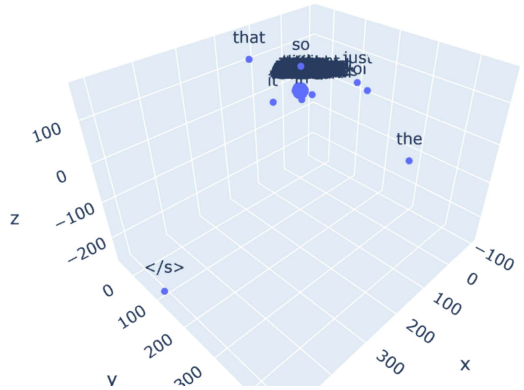
```
# Trực quan hóa 2D
plt.figure(figsize=(8, 8))
plt.scatter(word2vec_vectors_2d[:, 0], word2vec_vectors_2d[:, 1], s=10, alpha=0.7)
indices_word2vec = np.random.choice(range(len(word2vec_vocab)), 100, replace=False)
for i in indices_word2vec:
    plt.text(word2vec_vectors_2d[i, 0], word2vec_vectors_2d[i, 1], word2vec_vocab[i], fontsize=9)
plt.title(f'Trực quan hóa 2D cho {N_word2vec} từ Word2Vec')
plt.show()
```




```
# Trực quan hóa 3D
fig_word2vec = px.scatter_3d(
    x=word2vec_vectors_3d[:, 0],
    y=word2vec_vectors_3d[:, 1],
    z=word2vec_vectors_3d[:, 2],
    text=word2vec_vocab,
    title=f'Trực quan hóa 3D tương tác cho {N_word2vec} từ Word2Vec'
```

```
)
fig_word2vec.update_traces(marker=dict(size=3))
fig_word2vec.update_layout(width=800, height=500)
fig_word2vec.show()
```

Trực quan hóa 3D tương tác cho 500 từ Word2Vec

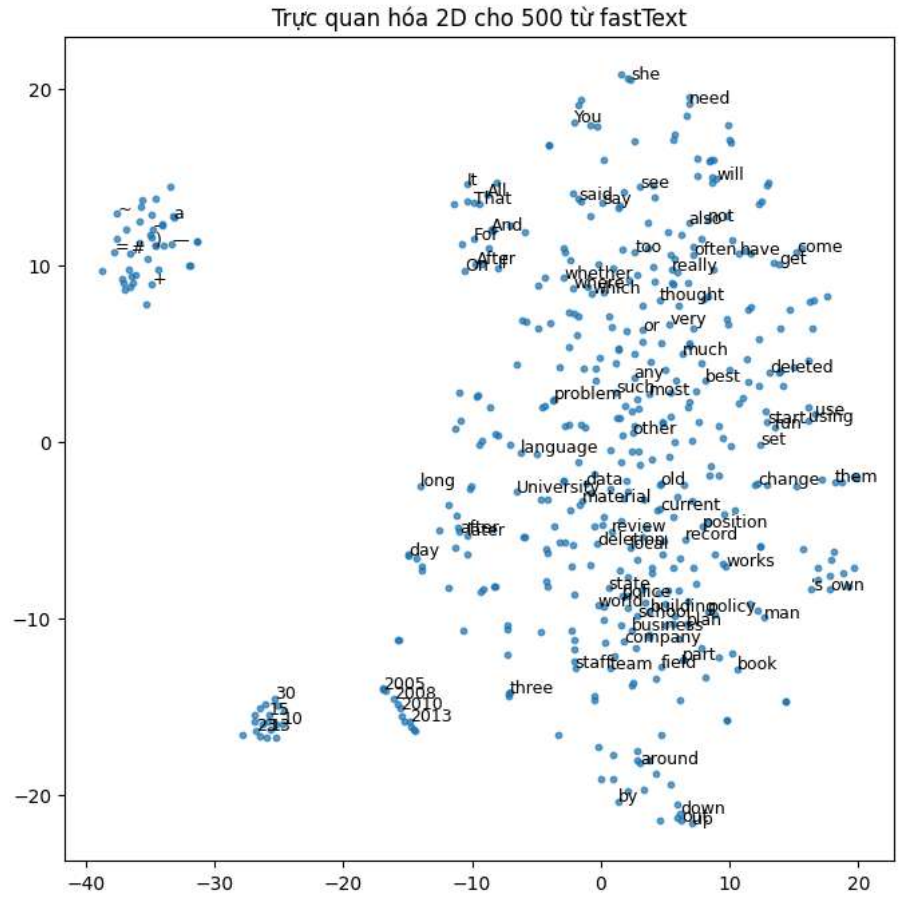


FASTTEXT

```
fasttext_model = api.load('fasttext-wiki-news-subwords-300')

N_fasttext = 500
fasttext_vocab = list(fasttext_model.index_to_key)[:N_fasttext]
fasttext_vectors = fasttext_model[fasttext_vocab]

tsne_fasttext_2d = TSNE(n_components=2, random_state=42, perplexity=30)
fasttext_vectors_2d = tsne_fasttext_2d.fit_transform(fasttext_vectors)

tsne_fasttext_3d = TSNE(n_components=3, random_state=42, perplexity=30)
fasttext_vectors_3d = tsne_fasttext_3d.fit_transform(fasttext_vectors)
```

```
# Trực quan hóa 2D
plt.figure(figsize=(8, 8))
plt.scatter(fasttext_vectors_2d[:, 0], fasttext_vectors_2d[:, 1], s=10, alpha=0.7)
```

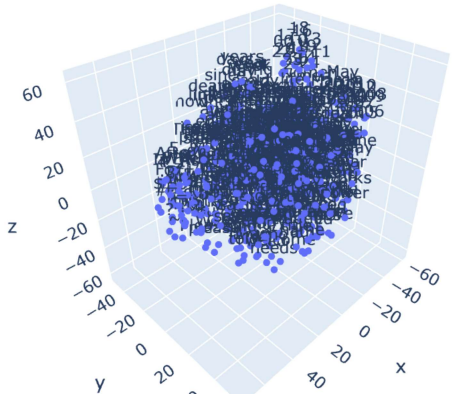
```
indices_fasttext = np.random.choice(range(len(fasttext_vocab)), 100, replace=False)
for i in indices_fasttext:
    plt.text(fasttext_vectors_2d[i, 0], fasttext_vectors_2d[i, 1], fasttext_vocab[i], fontsize=9)
plt.title(f'Trực quan hóa 2D cho {N_fasttext} từ fastText')
plt.show()
```




```
# Trực quan hóa 3D
fig_fasttext = px.scatter_3d(
    x=fasttext_vectors_3d[:, 0],
    y=fasttext_vectors_3d[:, 1],
    z=fasttext_vectors_3d[:, 2],
    text=fasttext_vocab,
    title=f'Trực quan hóa 3D tương tác cho {N_fasttext} từ fastText'
)
fig_fasttext.update_traces(marker=dict(size=3))
fig_fasttext.update_layout(width=800, height=500)
```

```
fig_fasttext.show()
```

Trực quan hóa 3D tương tác cho 500 từ fastText



TÌM KIẾM TƯƠNG ĐỒNG

```
K = 5
```

```
target_word_1 = 'dog'
print(f"--- Từ khóa: '{target_word_1}' (GloVe) ---")
try:
    similar_words_glove = glove_model.most_similar(target_word_1, topn=K)
    for word, score in similar_words_glove:
        print(f"- {word}: {score:.4f}")
except KeyError:
    print(f"Từ '{target_word_1}' không có trong từ điển.")

print("\n" + "="*50 + "\n")
```

```
--- Từ khóa: 'dog' (GloVe) ---
- cat: 0.8798
- dogs: 0.8344
- pet: 0.7450
- puppy: 0.7236
- horse: 0.7110
```

```
==================================================
```

```
target_word_2 = 'flower'
print(f"--- Từ khóa: '{target_word_2}' (Word2Vec) ---")
try:
    similar_words_word2vec = word2vec_model.most_similar(target_word_2, topn=K)
    for word, score in similar_words_word2vec:
        print(f"- {word}: {score:.4f}")
except KeyError:
    print(f"Từ '{target_word_2}' không có trong từ điển.")

print("\n" + "="*50 + "\n")
```

```
--- Từ khóa: 'flower' (Word2Vec) ---
- floral: 0.7494
- flowers: 0.7489
- roses: 0.6977
- orchid: 0.6929
- tulip: 0.6629

==================================================
```

```
target_word_3 = 'football'
print(f"--- Từ khóa: '{target_word_3}' (fastText) ---")
try:
    similar_words_fasttext = fasttext_model.most_similar(target_word_3, topn=K)
    for word, score in similar_words_fasttext:
        print(f"- {word}: {score:.4f}")
except KeyError:
    print(f"Từ '{target_word_3}' không có trong từ điển.")
```

```
--- Từ khóa: 'football' (fastText) ---
- footbal: 0.8377
- soccer: 0.8198
- football-: 0.8004
- non-football: 0.7719
- mini-football: 0.7673
```

NHẬN XÉT VÀ ĐÁNH GIÁ:

Nhận xét:

- Các cụm từ có ngữ nghĩa tương đồng thường nằm gần nhau trên mặt phẳng 2D.

- Biểu đồ của GloVe và Word2Vec cho thấy cấu trúc khá tương đồng: các cụm ngữ nghĩa được hình thành rõ, khoảng cách giữa các cụm khá tách biệt.
- Biểu đồ FastText, phân bố rải hơn, giúp mô hình hiểu tốt hơn các từ hiếm hoặc biến thể của từ, nhưng cũng khiến cụm trở nên loãng hơn.

Đánh giá:

- Việc sử dụng t-SNE 2D giúp quan sát được mối quan hệ ngữ nghĩa cơ bản giữa các từ, thể hiện được khả năng học của từng mô hình embedding.
- Tuy nhiên, t-SNE nhạy cảm với tham số (perplexity, learning rate), nên hình dạng cụm có thể thay đổi.