Setup: Install and import libraries

```
!pip install --upgrade gensim scikit-learn plotly
```

Hiện kết quả đã ẩn

```
import random
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import gensim.downloader as api
import plotly.express as px
import os
import time
```

Task 1: Using pretrained embeddings

```
#Load model
print("pretrained embeddings:glove-wiki-gigaword-50")
model = api.load("glove-wiki-gigaword-50")  # pretrained 50d embeddings
print(f"Loaded model. Vocab size: {len(model.index_to_key)}")
```

```
pretrained embeddings:glove-wiki-gigaword-50
Loaded model. Vocab size: 400000
```

```
#Vector of a word
word_vector = model['king']
print(f"Vector of 'python':\n{word_vector}")
print(f"vector size: {len(word_vector)}")
```

```
Vector of 'python':
[ 0.50451   0.68607  -0.59517  -0.022801  0.60046  -0.13498  -0.08813
  0.47377  -0.61798  -0.31012  -0.076666  1.493    -0.034189 -0.98173
  0.68229   0.81722  -0.51874  -0.31503  -0.55809   0.66421   0.1961
 -0.13495  -0.11476  -0.30344   0.41177  -2.223    -1.0756   -1.0783
 -0.34354   0.33505   1.9927   -0.04234  -0.64319   0.71125   0.49159
  0.16754   0.34344  -0.25663  -0.8523    0.1661    0.40102   1.1685
 -1.0137   -0.21585  -0.15155   0.78321  -0.91241  -1.6106   -0.64426
 -0.51042 ]
vector size: 50
```

```
# Similarity between 'king' và 'queen'
similarity_score = model.similarity('king', 'queen')
print(f"'king' - 'queen' score: {similarity_score:.4f}")

similarity_score_2 = model.similarity('dog', 'cat')
print(f"'dog' - 'cat' score: {similarity_score_2:.4f}")

similarity_score_3 = model.similarity('tank', 'horse')
print(f"'tank' - 'car': {similarity_score_3:.4f}")
```

```
'king' - 'queen' score: 0.7839
'dog' - 'cat' score: 0.9218
'tank' - 'car': 0.4111
```

```
# Top 10 similar word
most_similar_words = model.most_similar('king', topn=10)
print(f"Top 10 most similar to 'king':")
for word, score in most_similar_words:
    print(f"{word}: {score:.4f}")
```

```
Top 10 most similar to 'king':
prince: 0.8236
queen: 0.7839
ii: 0.7746
emperor: 0.7736
son: 0.7667
uncle: 0.7627
kingdom: 0.7542
```

```
throne: 0.7540
brother: 0.7492
ruler: 0.7434
```

Task 2: Word Embedding

Nhúng câu/văn bản bằng cách lấy trung bình vector từ

```
def embed_text_average(text, model):
    words = text.lower().split() # chữ thường, tách từ
    word_vectors = [model[word] for word in words if word in model.index_to_key]
    if not word_vectors:
        return np.zeros(model.vector_size) # vector 0
    return np.mean(word_vectors, axis=0)

# Ví dụ sử dụng hàm
sentence1 = "My name is Tim"
sentence2 = "My Tim is name"
sentence3 = "My.. uh, What?"

embedding1 = embed_text_average(sentence1, model)
embedding2 = embed_text_average(sentence2, model)
embedding3 = embed_text_average(sentence3, model)

# Frist 5
print(f"Embedding 1 (size {len(embedding1)}):\n{embedding1[:5]}...")
print(f"Embedding 2 (size {len(embedding2)}):\n{embedding2[:5]}...")
print(f"Embedding 3 (size {len(embedding3)}):\n{embedding3[:5]}...")

# similarity w/ cosine
from sklearn.metrics.pairwise import cosine_similarity

sim_1_2 = cosine_similarity([embedding1], [embedding2])[0][0]
sim_1_3 = cosine_similarity([embedding1], [embedding3])[0][0]

print(f"\n similarity 1 - 2: {sim_1_2:.4f}")
print(f" similarity 1 - 3: {sim_1_3:.4f}")
```

```
Embedding 1 (size 50):
[-0.06031001  0.47999752 -0.21364999  0.11847501  0.742     ]...
Embedding 2 (size 50):
[-0.06030999  0.47999752 -0.21365     0.118475    0.742     ]...
Embedding 3 (size 50):
[0. 0. 0. 0. 0.]...

 similarity 1 - 2: 1.0000
 similarity 1 - 3: 0.0000
```

Task3: Train Model on small data - GenSim

```
import gensim
from gensim.utils import simple_preprocess
import os

data_path = '/en_ewt-ud-dev.txt'

if not os.path.exists(data_path):
    print(f"Error: File not found")
else:
    print(f"Reading: {data_path}")
    sentences = []
    try:
        with open(data_path, 'r', encoding='utf-8') as f:
            for line in f:
                if line.strip():
                    sentences.append(simple_preprocess(line))
        print(f"Done processing. Lines: {len(sentences)}")
        print("\n5 example:")
        for i, sentence in enumerate(sentences[:5]):
            print(f"Câu {i+1}: {sentence}")
```

```
    except Exception as e:
        print(f"Error while reading: {e}")
```

```
Reading: /en_ewt-ud-dev.txt
Done processing. Lines: 1992

5 example:
Câu 1: ['from', 'the', 'ap', 'comes', 'this', 'story']
Câu 2: ['president', 'bush', 'on', 'tuesday', 'nominated', 'two', 'individuals', 'to', 'replace', 'retiring', 'jurists']
Câu 3: ['on', 'federal', 'courts', 'in', 'the', 'washington', 'area', 'bush', 'nominated', 'jennifer', 'anderson']
Câu 4: ['for', 'year', 'term', 'as', 'associate', 'judge', 'of', 'the', 'superior', 'court', 'of', 'the', 'district', 'of']
Câu 5: ['columbia', 'replacing', 'steffen', 'graae', 'bush', 'also', 'nominated', 'noel', 'anketell']
```

Task4: Train Model on big data - Spark

```
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.12/dist-packages (3.5.1)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.12/dist-packages (from pyspark) (0.10.9.7)
```

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Word2Vec Large Dataset Training") \
    .getOrCreate()
data_path_large = '/c4-train.00000-of-01024-30K.json'
large_df = spark.read.json(data_path_large) # read data

print("\nSample Data:")
large_df.show(5, truncate=50)
```

```
Sample Data:
+--------------------------------------------------+-------------------+--------------------------------------------------+
|                                              text|          timestamp|                                               url|
+--------------------------------------------------+-------------------+--------------------------------------------------+
|Beginners BBQ Class Taking Place in Missoula!\n...|2019-04-25T12:57:54Z|https://klyq.com/beginners-bbq-class-taking-pla...|
|Discussion in 'Mac OS X Lion (10.7)' started by...|2019-04-21T10:07:13Z|https://forums.macrumors.com/threads/restore-fr...|
|Foil plaid lycra and spandex shortall with meta...|2019-04-25T10:40:23Z|https://awishcometrue.com/Catalogs/Clearance/Tw...|
|How many backlinks per day for new site?\nDiscu...|2019-04-21T12:46:19Z|https://www.blackhatworld.com/seo/how-many-back...|
|The Denver Board of Education opened the 2017-1...|2019-04-20T14:33:21Z|            http://bond.dpsk12.org/category/news/|
+--------------------------------------------------+-------------------+--------------------------------------------------+
only showing top 5 rows
```

```
from pyspark.sql.functions import regexp_replace, split, lower

def preprocess_text_spark(text_col):
    text_col = lower(text_col) # to lowercase
    text_col = regexp_replace(text_col, "[^a-z0-9\\s]", "") # regex
    words = split(text_col, "\\s+") # split
    return words

large_df = large_df.withColumn("words", preprocess_text_spark(large_df["text"]))

print("DataFrame with preprocessed 'words' column:")
large_df.select("text", "words").show(5, truncate=50)
```

```
DataFrame with preprocessed 'words' column:
+--------------------------------------------------+--------------------------------------------------+
|                                              text|                                             words|
+--------------------------------------------------+--------------------------------------------------+
|Beginners BBQ Class Taking Place in Missoula!\n...|[beginners, bbq, class, taking, place, in, miss...|
|Discussion in 'Mac OS X Lion (10.7)' started by...|[discussion, in, mac, os, x, lion, 107, started...|
|Foil plaid lycra and spandex shortall with meta...|[foil, plaid, lycra, and, spandex, shortall, wi...|
|How many backlinks per day for new site?\nDiscu...|[how, many, backlinks, per, day, for, new, site...|
|The Denver Board of Education opened the 2017-1...|[the, denver, board, of, education, opened, the...|
+--------------------------------------------------+--------------------------------------------------+
only showing top 5 rows
```

```
from pyspark.ml.feature import Word2Vec

word2vec = Word2Vec(vectorSize=40, minCount=5, inputCol="words", outputCol="vectors")

print("Train Word2Vec with Spark MLlib on first 100 line")
```

```
word2Vec_model = word2vec.fit(large_df.limit(100)) # Chỉ lấy 100 dòng đầu
```

```
Đang huấn luyện mô hình Word2Vec bằng Spark MLlib trên 100 dòng đầu tiên...
Huấn luyện mô hình Word2Vec hoàn tất.
```

```
test_words_spark_subset = ["in", "and", "mac"]

for word in test_words_spark_subset:
    try:
        synonyms_spark_subset = word2Vec_model.findSynonyms(word, 5)
        print(f"[5] most similar wwith '{word}':")
        if synonyms_spark_subset.count() > 0:
            for row in synonyms_spark_subset.collect():
                print(f"- {row.word}: {row.similarity:.4f}")
        else:
             print(f"'{word}' have no similarity")

    except Exception as e:
        if "word not in vocabulary" in str(e):
            print(f"Từ '{word}' không có trong từ vựng của model Spark được huấn luyện trên 100 dòng đầu.")
        else:
            print(f"Lỗi khi tìm từ tương tự cho '{word}': {e}")

# example: 'story' vector
try:
    word_vector_spark_df_subset = word2Vec_model.getVectors().filter(f"word = 'story'")
    word_vector_spark_subset = word_vector_spark_df_subset.select("vector").first()
    if word_vector_spark_subset:
        print(f"\nVector 'story' (Spark, 100 line): {word_vector_spark_subset.vector[:10]}...") # Hiển thị 10 phần tử đầu
    else:
        print("\n'story' not in model Dict.")
except Exception as e:
     print(f"error when building 'story' vector: {e}")
```

```
[5] most similar wwith 'in':
- on: 0.9408
- ventures: 0.9355
- for: 0.9342
- resolution: 0.9063
- myanmar: 0.8975
[5] most similar wwith 'and':
- a: 0.8983
- in: 0.8927
- an: 0.8678
- half: 0.8437
- digital: 0.8356
[5] most similar wwith 'mac':
- hes: 0.9624
- outside: 0.9603
- important: 0.9599
- working: 0.9579
- hdd: 0.9563

Vector 'story' (Spark, 100 line): [-0.01263779 -0.01839844 -0.02680481  0.00380095 -0.00489168  0.0160823
  0.02787231  0.03469434  0.00259942 -0.04013837]...
```

Task5: Visualization

```
words_to_visualize = ['king', 'queen', 'man', 'woman', 'prince', 'princess', 'son', 'daughter', 'apple', 'banana', 'orange', 'grape
print(f"Words to visualize: {words_to_visualize}")
```

```
Words to visualize: ['king', 'queen', 'man', 'woman', 'prince', 'princess', 'son', 'daughter', 'apple', 'banana', 'orange', 'grape',
```

```
word_vectors_list = []
for word in words_to_visualize:
    if word in model.index_to_key:
        word_vectors_list.append(model[word])
    else:
        print(f"'{word}' not in vocabulary. Skipping.")

word_vectors_array = np.array(word_vectors_list)
print(f"Shape of word vectors array: {word_vectors_array.shape}")
```

```
Shape of word vectors array: (20, 50)
```

```
pca = PCA(n_components=2)
reduced_vectors_pca = pca.fit_transform(word_vectors_array)
print(f"Shape of PCA reduced vectors: {reduced_vectors_pca.shape}")
```

```
Shape of PCA reduced vectors: (20, 2)
```

```
tsne = TSNE(n_components=2, random_state=42, perplexity=5, n_iter_without_progress=250)
reduced_vectors_tsne = tsne.fit_transform(word_vectors_array)
print(f"Shape of t-SNE reduced vectors: {reduced_vectors_tsne.shape}")
```

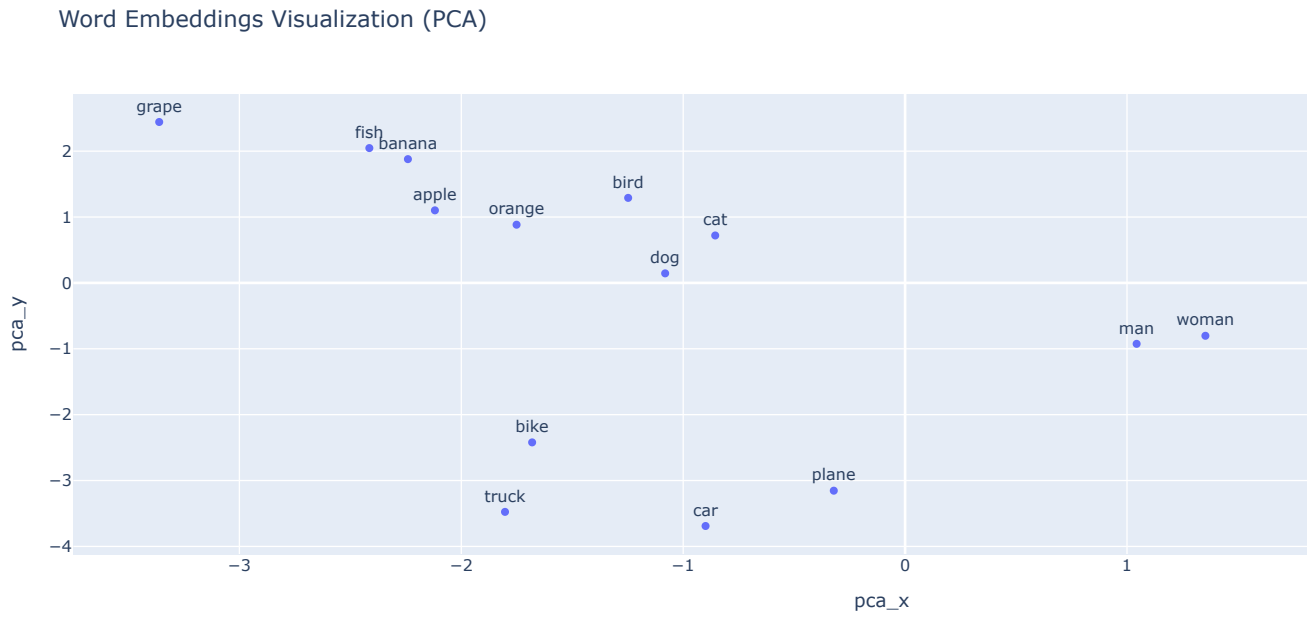
```
Shape of t-SNE reduced vectors: (20, 2)
```

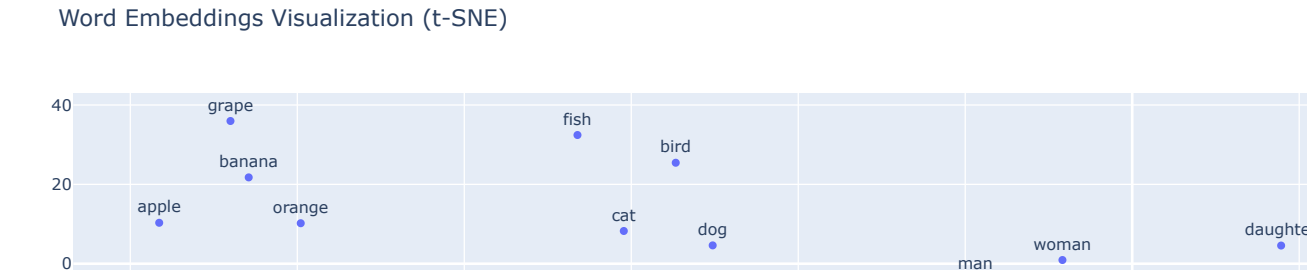
```
data = {
    'word': words_to_visualize,
    'pca_x': reduced_vectors_pca[:, 0],
    'pca_y': reduced_vectors_pca[:, 1],
    'tsne_x': reduced_vectors_tsne[:, 0],
    'tsne_y': reduced_vectors_tsne[:, 1]
}
embeddings_df = pd.DataFrame(data)
display(embeddings_df.head())
```

	word	pca_x	pca_y	tsne_x	tsne_y
0	king	2.669663	1.348948	56.601559	-2.829479
1	queen	2.601225	1.050173	51.747829	16.075314
2	man	1.043392	-0.925059	-18.781252	-4.061269
3	woman	1.353250	-0.802655	-8.357317	0.926782
4	prince	2.979707	0.748859	44.019314	1.624989

```
# Plot PCA results
fig_pca = px.scatter(embeddings_df, x='pca_x', y='pca_y', text='word', title='Word Embeddings Visualization (PCA)')
fig_pca.update_traces(textposition='top center')
fig_pca.show()

# Plot t-SNE results
fig_tsne = px.scatter(embeddings_df, x='tsne_x', y='tsne_y', text='word', title='Word Embeddings Visualization (t-SNE)')
fig_tsne.update_traces(textposition='top center')
fig_tsne.show()
```


Word Embeddings Visualization (PCA)
grape
fish
banana
apple
orange
bird
cat
dog
man
woman
bike
plane
truck
car
pca_y
pca_x
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1


Word Embeddings Visualization (t-SNE)
grape
fish
bird
banana
apple
orange
cat
dog
woman
man
40
20
0