In [ ]:
```
!pip install apache_beam

!pip install datasets

from datasets import load_dataset
dataset = load_dataset("wikipedia", "20220301.simple")
# check the first example of the training portion of the dataset:
print(dataset['train'][0])





from gensim.models import Word2Vec

# Define a function to preprocess the text
def preprocess(text):
    return text.lower().split()

# Define hyperparameters for the first model (skip-gram)
sg_1 = 1  # 1 for skip-gram, 0 for CBOW

window_1 = 5
min_count_1 = 5
workers_1 = 4

# Train the first Word2Vec model
sentences_1 = [preprocess(text) for text in dataset["train"]["text"]]
model_1 = Word2Vec(sentences_1, sg=sg_1, window=window_1, min_count=min_coun




# Define hyperparameters for the second model (CBOW)
sg_2 = 0  # 1 for skip-gram, 0 for CBOW

window_2 = 10
min_count_2 = 10
workers_2 = 8

# Train the second Word2Vec model
sentences_2 = [preprocess(text) for text in dataset["train"]["text"]]
model_2 = Word2Vec(sentences_2, sg=sg_2, window=window_2, min_count=min_coun


# Get the vocabulary and word vectors for each model
vocab_1 = list(model_1.wv.key_to_index.keys())
word_vectors_1 = model_1.wv[vocab_1]

vocab_2 = list(model_2.wv.key_to_index.keys())
word_vectors_2 = model_2.wv[vocab_2]
```

```
vocab_1 = list(model_1.wv.key_to_index.keys())
word_vectors_1 = model_1.wv[vocab_1]

word_vectors_1

import numpy as np

# Query 1: Get the top 10 most similar words to 'piano' using the first mode
similar_words_1 = model_1.wv.most_similar('piano', topn=10)
print(f"Similar words to 'piano' using model 1: {similar_words_1}")

import numpy as np

# Query 1: Get the top 10 most similar words to 'piano' using the first mode
similar_words_1 = model_2.wv.most_similar('piano', topn=10)
print(f"Similar words to 'piano' using model 1: {similar_words_1}")

Similar words to 'piano' using model 1: [('violin', 0.8726906776428223), ('c
                                         ('concerto', 0.856177568435669), ('
                                         ('concertos', 0.8392475843429565),
                                         ('clarinet', 0.8331102728843689), (
                                         ('oboe', 0.8217369318008423), ('cla



# Query 3: Get the cosine similarity between 'cat' and 'dog' using the first
similarity_1 = model_1.wv.similarity('cat', 'dog')
print(f"Similarity between 'cat' and 'dog' using model 1: {similarity_1}")

# Query 4: Get the word that is closest to the vector 'king' - 'man' + 'woma
vector = model_1.wv['king'] - model_1.wv['man'] + model_1.wv['woman']
similar_word_2 = model_1.wv.similar_by_vector(vector, topn=1)
print(f"Word closest to 'king' - 'man' + 'woman' using model 1: {similar_wor


vector = (model_1.wv['doctor'] - model_1.wv['man']) + (model_1.wv['surgeon']
similar_word_2 = model_1.wv.similar_by_vector(vector, topn=1)
print(f"Word closest to 'king' - 'man' + 'woman' using model 1: {similar_wor

model_1.wv.most_similar(positive=['king', 'woman'],
                        negative=['man'])

model_1.wv.most_similar(positive=['sister', 'he'],
                        negative=['she'])

model_2.wv.most_similar(positive=['sister', 'he'],
                        negative=['she'])

model_1.wv.most_similar(positive=['nurse', 'he'],
                        negative=['she'])
```

```
model_2.wv.most_similar(positive=['boss', 'he'],
                        negative=['she'])

model_1.wv.most_similar(positive=['surgeon', 'she'],
                        negative=['he'])

model_2.wv.most_similar(positive=['boss', 'she'],
                        negative=['he'])

model_glove.most_similar(positive=['brave', 'she'],
                        negative=['he'])

model_glove.most_similar(positive=['sweet', 'he'],
                        negative=['she'])



vector

import gensim.downloader as api

model_glove = api.load("glove-wiki-gigaword-100")

# find similarity
model_glove.most_similar(positive=['piano'],topn=10)



# Define the queries
queries = {
    'piano': "Words similar to 'piano':",
    'happy': "Words similar to 'happy':",
    'cat_dog': "Cosine similarity between 'cat' and 'dog':",
    'king_man_woman': "Word closest to 'king' - 'man' + 'woman':",
    'computer': "Vector representation of 'computer':"
}

# Run the queries on each model
for name, model in [('model_1', model_1),('GloVe', model_glove)]:
    print(f"Results for {name}:")
    for query in queries:
        if query == 'piano' or query == 'happy':
            similar_words = model.most_similar(query, topn=10)
            print(f"{queries[query]} {[word[0] for word in similar_words]}")
        elif query == 'cat_dog':
            similarity = model.similarity('cat', 'dog')
            print(f"{queries[query]} {similarity:.4f}")
        elif query == 'king_man_woman':
            vector = model['king'] - model['man'] + model['woman']
            similar_word = model.similar_by_vector(vector, topn=1)
            print(f"{queries[query]} {similar_word[0][0]}")
        elif query == 'computer':
                vector = model['computer']
                print(f"{queries[query]} {vector}")
        print()




# Define the queries
queries = {
    'piano': "Words similar to 'piano':",
    'happy': "Words similar to 'happy':",
    'cat_dog': "Cosine similarity between 'cat' and 'dog':",
    'king_man_woman': "Word closest to 'king' - 'man' + 'woman':",
    'computer': "Vector representation of 'computer':"
}

# Run the queries on each model
for name, model in [('Model 2', model_2)]:
    print(f"Results for {name}:")
    for query in queries:
        if query == 'piano' or query == 'happy':
            similar_words = model.wv.most_similar(query, topn=10)
            print(f"{queries[query]} {[word[0] for word in similar_words]}")
        elif query == 'cat_dog':
            similarity = model.wv.similarity('cat', 'dog')
            print(f"{queries[query]} {similarity:.4f}")
        elif query == 'king_man_woman':
            vector = model.wv['king'] - model.wv['man'] + model.wv['woman']
            similar_word = model.wv.similar_by_vector(vector, topn=1)
            print(f"{queries[query]} {similar_word[0][0]}")
        elif query == 'computer':
            vector = model.wv['computer']
            print(f"{queries[query]} {vector}")
    print()
```