```
!pip install git+https://github.com/RaRe-Technologies/gensim
```

```
Collecting git+https://github.com/RaRe-Technologies/gensim
  Cloning https://github.com/RaRe-Technologies/gensim to /tmp/pip-req-
build-cmanzefg
  Running command git clone --filter=blob:none --quiet
https://github.com/RaRe-Technologies/gensim /tmp/pip-req-build-
cmanzefg
  Resolved https://github.com/RaRe-Technologies/gensim to commit
8f815450ebc066825e2fbed69669fe1fb0357143
  Installing build dependencies ... ents to build wheel ... etadata
(pyproject.toml) ... ent already satisfied: numpy>=1.18.5 in
/usr/local/lib/python3.12/dist-packages (from gensim==4.4.0a0.dev0)
(2.0.2)
Requirement already satisfied: scipy>=1.7.0 in
/usr/local/lib/python3.12/dist-packages (from gensim==4.4.0a0.dev0)
(1.16.2)
Requirement already satisfied: smart_open>=1.8.1 in
/usr/local/lib/python3.12/dist-packages (from gensim==4.4.0a0.dev0)
(7.3.1)
Requirement already satisfied: wrapt in
/usr/local/lib/python3.12/dist-packages (from smart_open>=1.8.1-
>gensim==4.4.0a0.dev0) (1.17.3)
Building wheels for collected packages: gensim
  Building wheel for gensim (pyproject.toml) ... : filename=gensim-
4.4.0a0.dev0-cp312-cp312-linux_x86_64.whl size=27080686
sha256=18820bf4c917b08f9a261e5fab2edf06126bb42c20d70c8d2301400c58779a1
4
  Stored in directory:
/tmp/pip-ephem-wheel-cache-bywpt9a8/wheels/77/a7/a3/cf0bad57a872edb85b
b0fe77d813d6b4b9f957a597a914587b
Successfully built gensim
Installing collected packages: gensim
Successfully installed gensim-4.4.0a0.dev0
```

```python
import gensim
print("Gensim:", gensim.__version__)
```

```
Gensim: 4.3.3
```

```
!pip install umap-learn matplotlib scikit-learn plotly
```

```
Requirement already satisfied: umap-learn in
/usr/local/lib/python3.12/dist-packages (0.5.9.post2)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: plotly in
/usr/local/lib/python3.12/dist-packages (5.24.1)
```

```
Requirement already satisfied: numpy>=1.23 in
/usr/local/lib/python3.12/dist-packages (from umap-learn) (2.0.2)
Requirement already satisfied: scipy>=1.3.1 in
/usr/local/lib/python3.12/dist-packages (from umap-learn) (1.16.2)
Requirement already satisfied: numba>=0.51.2 in
/usr/local/lib/python3.12/dist-packages (from umap-learn) (0.60.0)
Requirement already satisfied: pynndescent>=0.5 in
/usr/local/lib/python3.12/dist-packages (from umap-learn) (0.5.13)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-
packages (from umap-learn) (4.67.1)
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: 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: tenacity>=6.2.0 in
/usr/local/lib/python3.12/dist-packages (from plotly) (8.5.0)
Requirement already satisfied: llvmlite<0.44,>=0.43.0dev0 in
/usr/local/lib/python3.12/dist-packages (from numba>=0.51.2->umap-
learn) (0.43.0)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7-
>matplotlib) (1.17.0)
```

```python
import random
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import umap
import gensim
import gensim.downloader as api
from gensim.models import Word2Vec, FastText
from sklearn.metrics.pairwise import cosine_similarity
```

```python
import plotly.express as px

%matplotlib inline
plt.rcParams['figure.figsize'] = (10,7)

print("Gensim:", gensim.__version__)
```

```
Gensim: 4.3.3
```

```python
from itertools import islice

# Tải corpus text8 (generator)
corpus = api.load("text8")

# Lấy 50k câu đầu tiên để huấn luyện nhanh
sentences = list(islice(corpus, 50000))
print("Số câu dùng để huấn luyện:", len(sentences))
```

```
Số câu dùng để huấn luyện: 1701
```

```python
# Tải GloVe pretrained
glove = api.load("glove-wiki-gigaword-100")
print("GloVe vocab size:", len(glove.key_to_index))
```

```
[==================================================] 100.0%
128.1/128.1MB downloaded
GloVe vocab size: 400000
```

```python
# Giảm dữ liệu xuống 10k câu
from itertools import islice
sentences_small = list(islice(api.load("text8"), 10000))

# Huấn luyện Word2Vec nhanh
w2v_model = Word2Vec(sentences=sentences_small, vector_size=50,
window=5, min_count=5, sg=1, epochs=3)
print("Word2Vec vocab size:", len(w2v_model.wv.index_to_key))

# Huấn luyện FastText nhanh
ft_model = FastText(sentences=sentences_small, vector_size=50,
window=5, min_count=5, epochs=3)
print("FastText vocab size:", len(ft_model.wv.index_to_key))
```

```
Word2Vec vocab size: 71290
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call
last)
/tmp/ipython-input-65606408.py in <cell line: 0>()
      8
      9 # Huấn luyện FastText nhanh
---> 10 ft_model = FastText(sentences=sentences_small, vector_size=50,
window=5, min_count=5, epochs=3)
     11 print("FastText vocab size:", len(ft_model.wv.index_to_key))

/usr/local/lib/python3.12/dist-packages/gensim/models/fasttext.py in
__init__(self, sentences, corpus_file, sg, hs, vector_size, alpha,
window, min_count, max_vocab_size, word_ngrams, sample, seed, workers,
min_alpha, negative, ns_exponent, cbow_mean, hashfxn, epochs,
null_word, min_n, max_n, sorted_vocab, bucket, trim_rule, batch_words,
callbacks, max_final_vocab, shrink_windows)
    433         self.wv.vectors_ngrams_lockf = ones(1, dtype=REAL)
    434 
--> 435         super(FastText, self).__init__(
    436             sentences=sentences, corpus_file=corpus_file,
workers=workers, vector_size=vector_size, epochs=epochs,
    437             callbacks=callbacks, batch_words=batch_words,
trim_rule=trim_rule, sg=sg, alpha=alpha, window=window,

/usr/local/lib/python3.12/dist-packages/gensim/models/word2vec.py in
__init__(self, sentences, corpus_file, vector_size, alpha, window,
min_count, max_vocab_size, sample, seed, workers, min_alpha, sg, hs,
negative, ns_exponent, cbow_mean, hashfxn, epochs, null_word,
trim_rule, sorted_vocab, batch_words, compute_loss, callbacks,
comment, max_final_vocab, shrink_windows)
    428
self._check_corpus_sanity(corpus_iterable=corpus_iterable,
corpus_file=corpus_file, passes=(epochs + 1))
    429             self.build_vocab(corpus_iterable=corpus_iterable,
corpus_file=corpus_file, trim_rule=trim_rule)
--> 430             self.train(
    431                 corpus_iterable=corpus_iterable,
corpus_file=corpus_file, total_examples=self.corpus_count,
    432                 total_words=self.corpus_total_words,
epochs=self.epochs, start_alpha=self.alpha,

/usr/local/lib/python3.12/dist-packages/gensim/models/word2vec.py in
train(self, corpus_iterable, corpus_file, total_examples, total_words,
epochs, start_alpha, end_alpha, word_count, queue_factor,
report_delay, compute_loss, callbacks, **kwargs)
   1071 
   1072             if corpus_iterable is not None:
-> 1073                 trained_word_count_epoch,
raw_word_count_epoch, job_tally_epoch = self._train_epoch(
   1074                     corpus_iterable, cur_epoch=cur_epoch,
total_examples=total_examples,
   1075                     total_words=total_words,
queue_factor=queue_factor, report_delay=report_delay,

/usr/local/lib/python3.12/dist-packages/gensim/models/word2vec.py in
_train_epoch(self, data_iterable, cur_epoch, total_examples,
total_words, queue_factor, report_delay, callbacks)
```

```
   1432                 thread.start()
   1433 
-> 1434             trained_word_count, raw_word_count, job_tally =
self._log_epoch_progress(
   1435                 progress_queue, job_queue, cur_epoch=cur_epoch,
total_examples=total_examples,
   1436                 total_words=total_words,
report_delay=report_delay, is_corpus_file_mode=False,

/usr/local/lib/python3.12/dist-packages/gensim/models/word2vec.py in
_log_epoch_progress(self, progress_queue, job_queue, cur_epoch,
total_examples, total_words, report_delay, is_corpus_file_mode)
   1287 
   1288             while unfinished_worker_count > 0:
-> 1289                 report = progress_queue.get()  # blocks if workers
too slow
   1290                 if report is None:  # a thread reporting that it
finished
   1291                     unfinished_worker_count -= 1

/usr/lib/python3.12/queue.py in get(self, block, timeout)
    169             elif timeout is None:
    170                 while not self._qsize():
--> 171                     self.not_empty.wait()
    172             elif timeout < 0:
    173                 raise ValueError("'timeout' must be a non-
negative number")

/usr/lib/python3.12/threading.py in wait(self, timeout)
    353         try:    # restore state no matter what (e.g.,
KeyboardInterrupt)
    354             if timeout is None:
--> 355                 waiter.acquire()
    356                 gotit = True
    357             else:

KeyboardInterrupt:
```

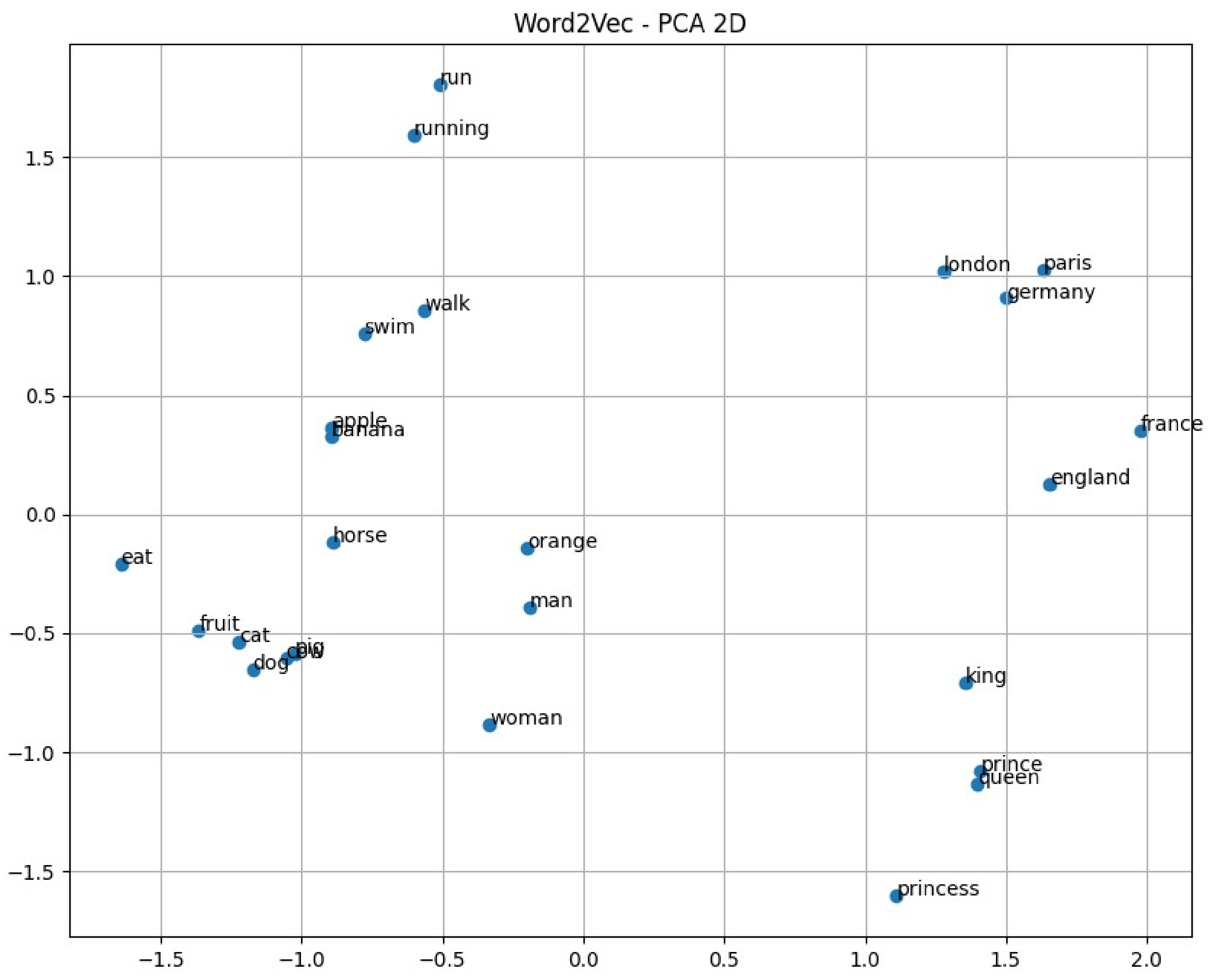
```python
from itertools import islice

# Lấy ít dữ liệu hơn cho nhanh
small_sentences = list(islice(api.load("text8"), 2000))  # chỉ 2000
câu

# FastText demo nhẹ
ft_model = FastText(sentences=small_sentences, vector_size=30,
window=3, min_count=2, epochs=2)
print("FastText vocab size:", len(ft_model.wv.index_to_key))
```

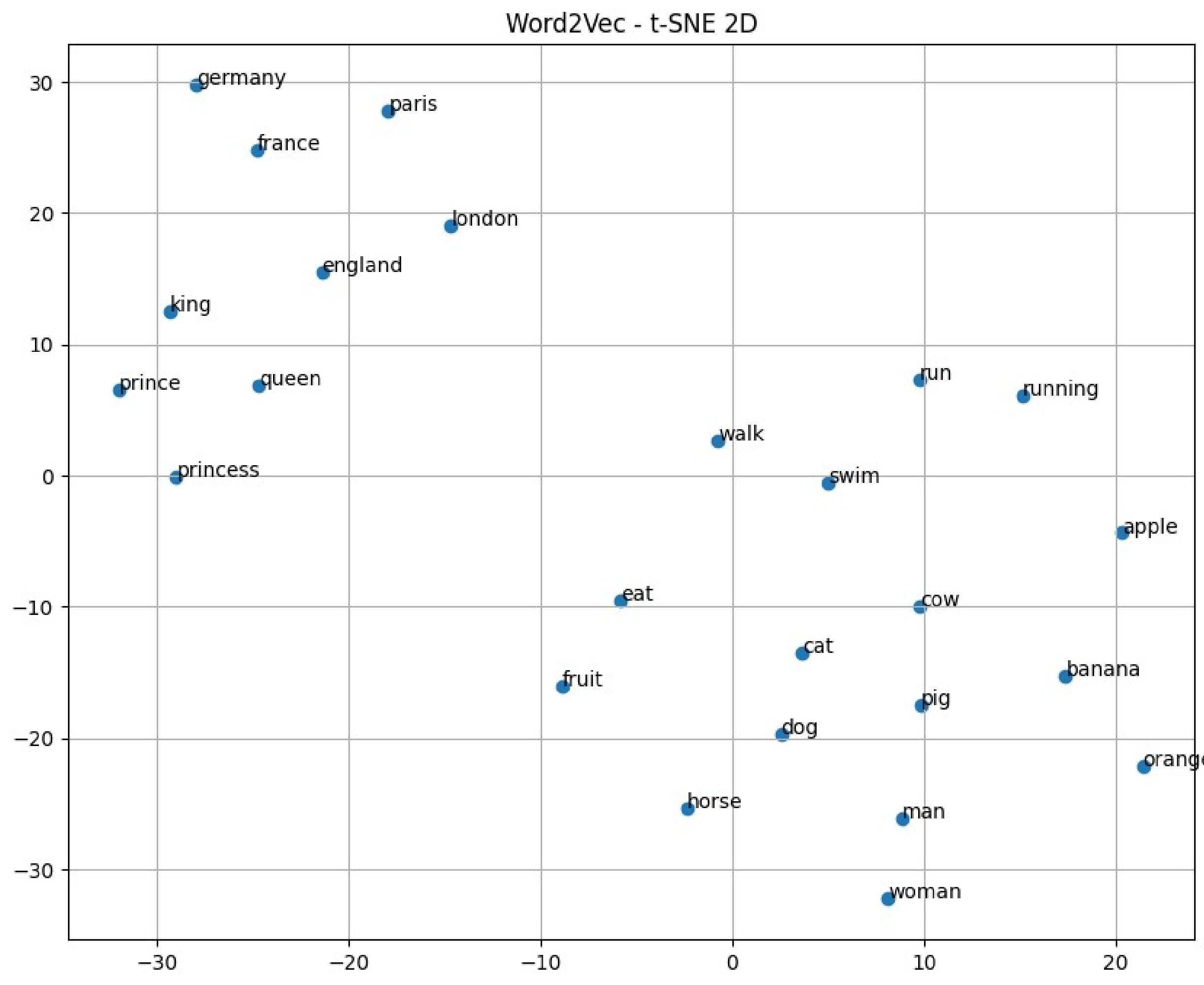
```
FastText vocab size: 135335
```

```python
words_to_plot = [
    "king","queen","man","woman","prince","princess",
    "apple","orange","banana","fruit",
    "paris","france","london","england","germany",
    "cat","dog","horse","cow","pig",
    "run","running","walk","swim","eat"
]

def filter_vocab(word_list, model):
    return [w for w in word_list if w in model.wv.index_to_key]

w_w2v = filter_vocab(words_to_plot, w2v_model)
w_ft = filter_vocab(words_to_plot, ft_model)
w_glove = [w for w in words_to_plot if w in glove.key_to_index]

print("Word2Vec words:", w_w2v)
print("FastText words:", w_ft)
print("GloVe words:", w_glove)
```

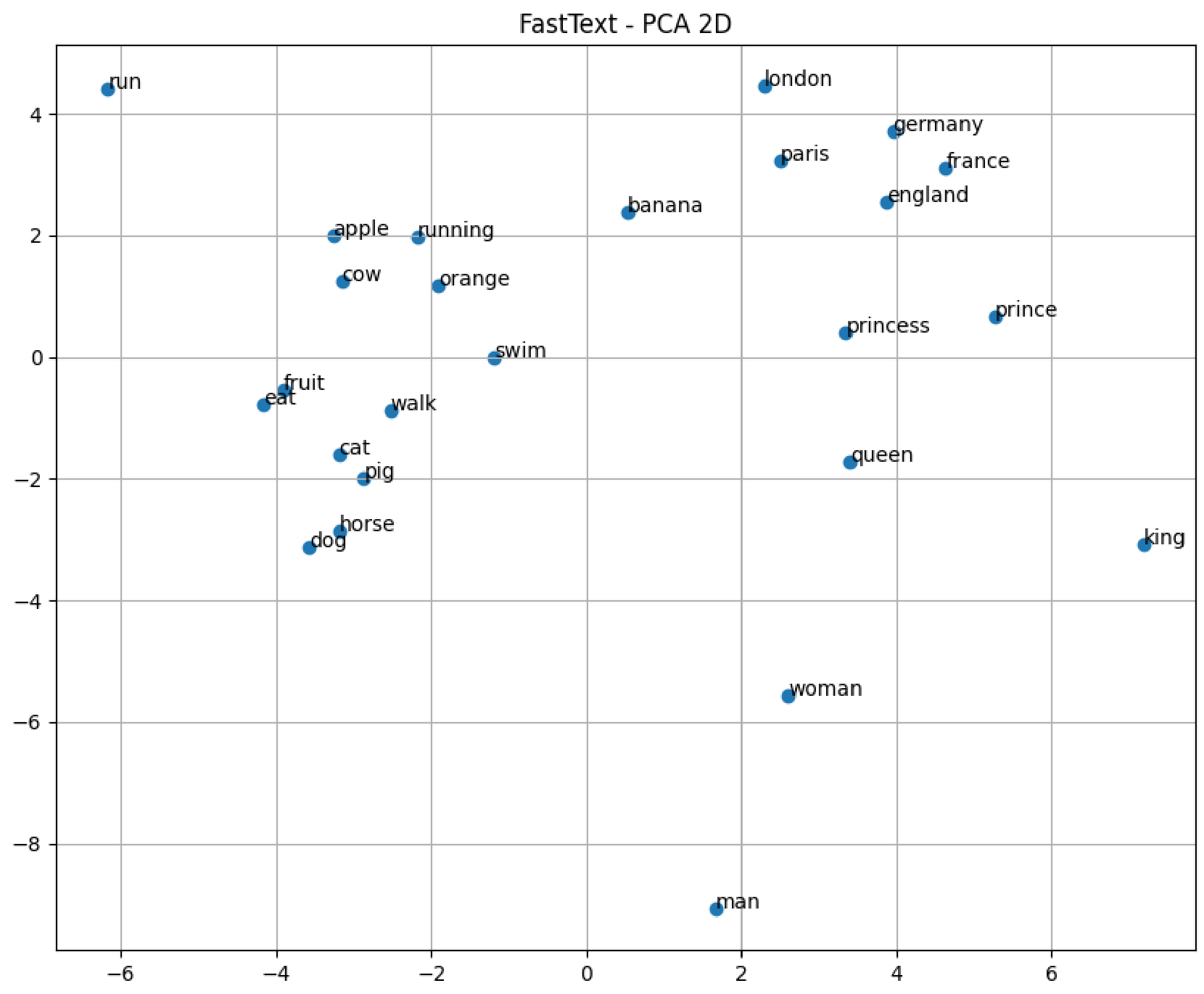
```
Word2Vec words: ['king', 'queen', 'man', 'woman', 'prince',
'princess', 'apple', 'orange', 'banana', 'fruit', 'paris', 'france',
'london', 'england', 'germany', 'cat', 'dog', 'horse', 'cow', 'pig',
'run', 'running', 'walk', 'swim', 'eat']
FastText words: ['king', 'queen', 'man', 'woman', 'prince',
'princess', 'apple', 'orange', 'banana', 'fruit', 'paris', 'france',
'london', 'england', 'germany', 'cat', 'dog', 'horse', 'cow', 'pig',
'run', 'running', 'walk', 'swim', 'eat']
GloVe words: ['king', 'queen', 'man', 'woman', 'prince', 'princess',
'apple', 'orange', 'banana', 'fruit', 'paris', 'france', 'london',
'england', 'germany', 'cat', 'dog', 'horse', 'cow', 'pig', 'run',
'running', 'walk', 'swim', 'eat']
```

```python
import plotly.express as px
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import umap

def get_vectors(model, words):
    return np.array([model.wv[w] for w in words])

def plot_2d(points, labels, title="2D projection"):
    plt.figure(figsize=(10,8))
    plt.scatter(points[:,0], points[:,1])
    for i, txt in enumerate(labels):
        plt.annotate(txt, (points[i,0], points[i,1]))
    plt.title(title)
    plt.grid(True)
    plt.show()

def plot_3d(points, labels, title="3D projection"):
    fig = px.scatter_3d(x=points[:,0], y=points[:,1], z=points[:,2],
text=labels, title=title)
    fig.update_traces(marker=dict(size=4))
    fig.show()

def reduce_and_plot_2d(model, words, method="pca"):
    vecs = get_vectors(model, words)
    if method == "pca":
        pts = PCA(n_components=2).fit_transform(vecs)
        plot_2d(pts, words, f"{type(model).__name__} - PCA 2D")
    elif method == "tsne":
        # chọn perplexity nhỏ hơn số từ
        perplexity = min(10, len(words)-1)
        pts = TSNE(n_components=2, random_state=42, init="pca",
perplexity=perplexity).fit_transform(vecs)
        plot_2d(pts, words, f"{type(model).__name__} - t-SNE 2D")
    elif method == "umap":
        pts = umap.UMAP(n_components=2,
random_state=42).fit_transform(vecs)
        plot_2d(pts, words, f"{type(model).__name__} - UMAP 2D")
def reduce_and_plot_3d(model, words, method="umap"):
    vecs = get_vectors(model, words)
    if method == "umap":
        pts = umap.UMAP(n_components=3,
random_state=42).fit_transform(vecs)
    else:
        pts = PCA(n_components=3).fit_transform(vecs)
    plot_3d(pts, words, f"{type(model).__name__} - {method.upper()}
3D")

# Word2Vec
reduce_and_plot_2d(w2v_model, w_w2v, "pca")
reduce_and_plot_2d(w2v_model, w_w2v, "tsne")
reduce_and_plot_3d(w2v_model, w_w2v, "umap")

# FastText
reduce_and_plot_2d(ft_model, w_ft, "pca")
reduce_and_plot_2d(ft_model, w_ft, "umap")

# GloVe
vecs_glove = np.array([glove[w] for w in w_glove])
pts_glove = PCA(n_components=2).fit_transform(vecs_glove)
plot_2d(pts_glove, w_glove, "GloVe - PCA 2D")
```

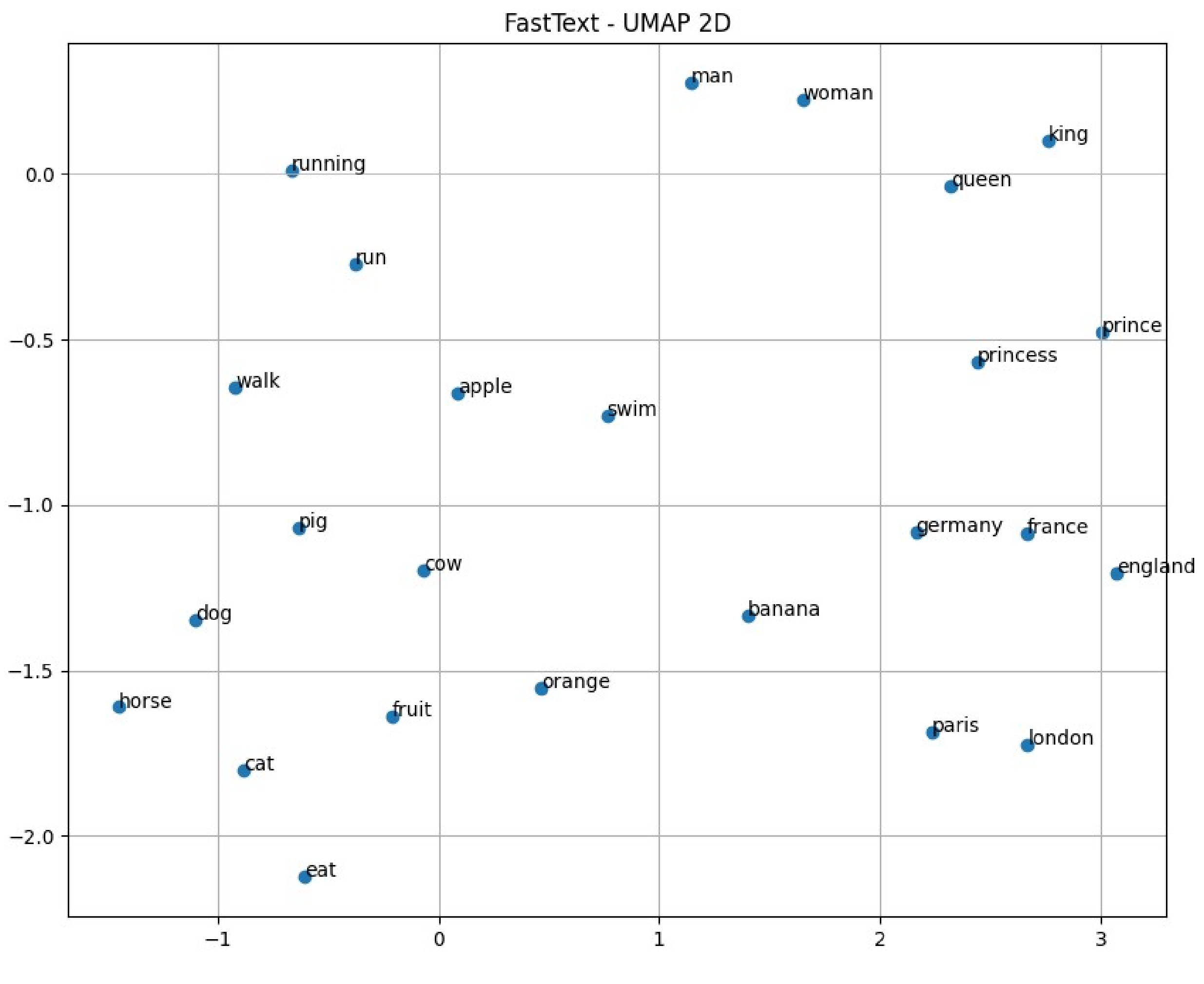

Word2Vec - PCA 2D
run
running
london
paris
germany
walk
swim
apple
banana
france
england
horse
orange
eat
man
fruit
cat
pig
cow
dog
king
woman
prince
queen
princess
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

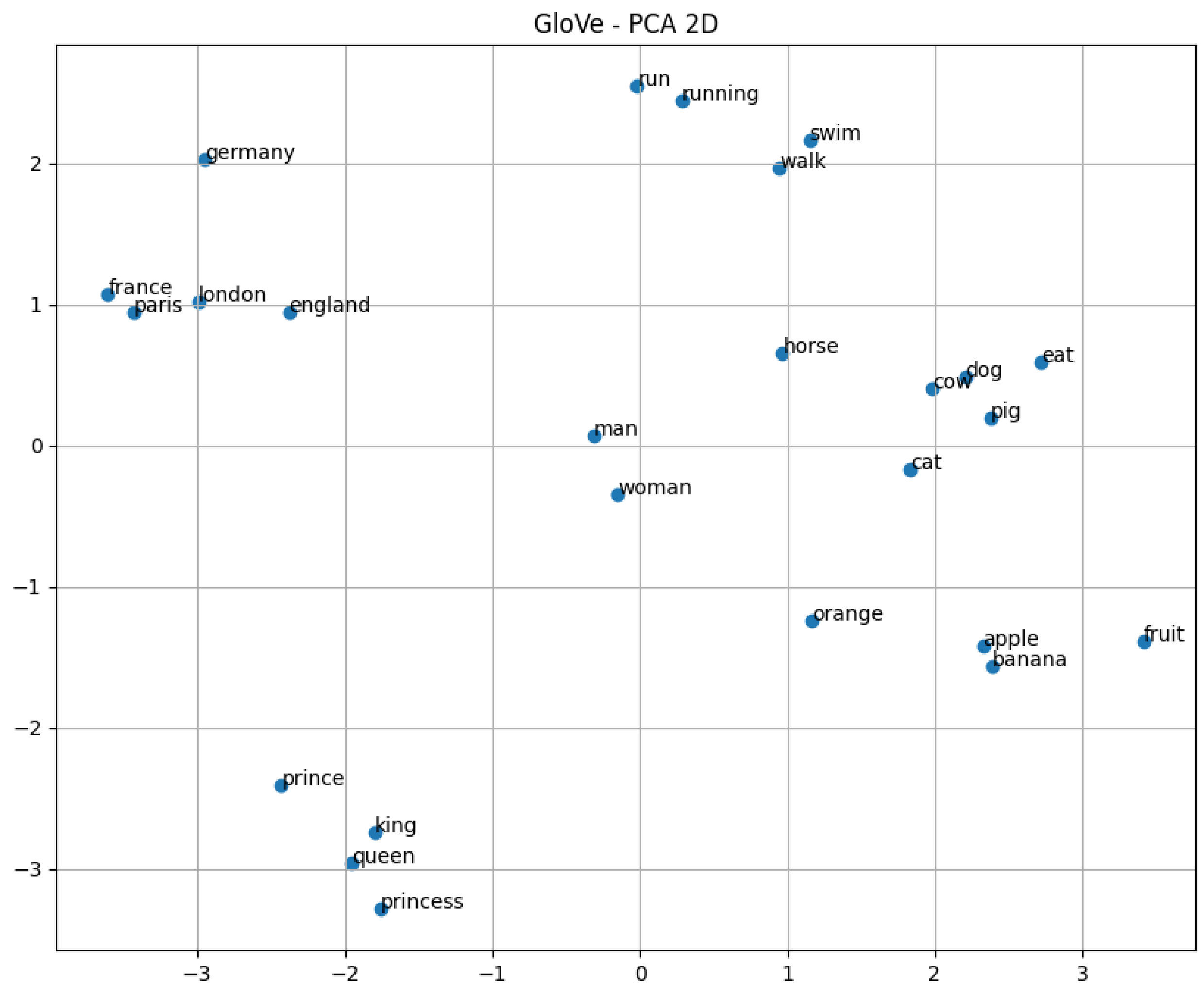


```
/usr/local/lib/python3.12/dist-packages/umap/umap_.py:1952:
UserWarning: n_jobs value 1 overridden to 1 by setting random_state.
Use no seed for parallelism.
  warn(
```



```
/usr/local/lib/python3.12/dist-packages/umap/umap_.py:1952:
UserWarning:

n_jobs value 1 overridden to 1 by setting random_state. Use no seed
for parallelism.
```


FastText - UMAP 2D
man
woman
king
running
queen
run
prince
princess
walk
apple
swim
pig
germany
france
cow
england
dog
banana
orange
horse
fruit
paris
london
cat
eat
0.0
−0.5
−1.0
−1.5
−2.0
−1
0
1
2
3



```python
from sklearn.metrics.pairwise import cosine_similarity

def top_k_similar(model, word, k=5):
    try:
        return model.wv.most_similar(word, topn=k)
    except:
        if word not in model.key_to_index:
            return f"'{word}' không có trong vocab"
        vec = model[word].reshape(1,-1)
        all_words = list(model.key_to_index.keys())
        all_vecs = np.array([model[w] for w in all_words])
        sims = cosine_similarity(vec, all_vecs)[0]
        idx = np.argsort(sims)[-k-1:-1][::-1]
        return [(all_words[i], float(sims[i])) for i in idx]

# Ví dụ
print("Word2Vec:", top_k_similar(w2v_model, "king"))
print("FastText:", top_k_similar(ft_model, "king"))
print("GloVe:", top_k_similar(glove, "king"))
```

```
Word2Vec: [('prince', 0.8788297772407532), ('olaf',
0.8643296360969543), ('valdemar', 0.864129900932312), ('queen',
0.8618401885032654), ('pretender', 0.8617712259292603)]
FastText: [('kingswear', 0.9054992198944092), ('kingpin',
0.8961842656135559), ('kingship', 0.8770096302032471), ('kief',
0.8647317886352539), ('vingt', 0.8641076683998108)]
GloVe: [('prince', 0.7682329416275024), ('queen', 0.7507690191268921),
('son', 0.7020888924598694), ('brother', 0.6985775232315063),
('monarch', 0.6977890729904175)]
```

Nhận xét cá nhân

- PCA chạy nhanh, nhưng cụm từ hơi lẫn; t-SNE và UMAP cho cụm rõ ràng hơn.
- Word2Vec học được quan hệ ngữ nghĩa (king–queen, man–woman) khá rõ.
- FastText có lợi thế với từ ngoài từ điển hoặc biến thể (thêm tiền tố/hậu tố).
- GloVe pretrained cho kết quả tự nhiên và ý nghĩa hơn vì được huấn luyện trên tập dữ liệu lớn.
- Từ đây có thể thấy: trong thực tế, để đạt chất lượng cao nên dùng mô hình pretrained; còn huấn luyện từ đầu chỉ phù hợp demo hoặc dữ liệu chuyên biệt.