Dense Retrieval

Setup

Load needed API keys and relevant Python libaries.

In [ ]:

```
# !pip install cohere
# !pip install weaviate-client Annoy
```

In [ ]:

```
import os
from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv()) # read local .env file
```

In [ ]:

```
import cohere
co = cohere.Client(os.environ['COHERE_API_KEY'])
```

In [ ]:

```
import weaviate
auth_config = weaviate.auth.AuthApiKey(
    api_key=os.environ['WEAVIATE_API_KEY'])
```

In [ ]:

```
client = weaviate.Client(
    url=os.environ['WEAVIATE_API_URL'],
    auth_client_secret=auth_config,
    additional_headers={
        "X-Cohere-Api-Key": os.environ['COHERE_API_KEY'],
    }
)
client.is_ready() #check if True
```

Part 1: Vector Database for semantic Search

In [ ]:

```
def dense_retrieval(query,
                    results_lang='en',
                    properties = ["text", "title", "url", "views", "lang", "_addit
                    num_results=5):

    nearText = {"concepts": [query]}

    # To filter by language
    where_filter = {
    "path": ["lang"],
    "operator": "Equal",
    "valueString": results_lang
    }
    response = (
        client.query
        .get("Articles", properties)
        .with_near_text(nearText)
        .with_where(where_filter)
        .with_limit(num_results)
        .do()
    )

    result = response['data']['Get']['Articles']

    return result
```

In [ ]:

```
from utils import print_result
```

Bacic Query

In [ ]:

```
query = "Who wrote Hamlet?"
dense_retrieval_results = dense_retrieval(query)
print_result(dense_retrieval_results)
```

Medium Query

In [ ]:

```
query = "What is the capital of Canada?"
dense_retrieval_results = dense_retrieval(query)
print_result(dense_retrieval_results)
```

In [ ]:

```
from utils import keyword_search

query = "What is the capital of Canada?"
keyword_search_results = keyword_search(query, client)
print_result(keyword_search_results)
```

Complicated Query

In [ ]:

```
from utils import keyword_search

query = "Tallest person in history?"
keyword_search_results = keyword_search(query, client)
print_result(keyword_search_results)
```

In [ ]:

```
query = "Tallest person in history"
dense_retrieval_results = dense_retrieval(query)
print_result(dense_retrieval_results)
```

In [ ]:

```
query = "أطول رجل في التاريخ"
dense_retrieval_results = dense_retrieval(query)
print_result(dense_retrieval_results)
```

In [ ]:

```
query = "film about a time travel paradox"
dense_retrieval_results = dense_retrieval(query)
print_result(dense_retrieval_results)
```

Part 2: Building Semantic Search from Scratch

Get the text archive:

In [ ]:

```
from annoy import AnnoyIndex
import numpy as np
import pandas as pd
import re
```

In [ ]:

```
text = """
Interstellar is a 2014 epic science fiction film co-written, directed, and produce
It stars Matthew McConaughey, Anne Hathaway, Jessica Chastain, Bill Irwin, Ellen B
Set in a dystopian future where humanity is struggling to survive, the film follow

Brothers Christopher and Jonathan Nolan wrote the screenplay, which had its origin
Caltech theoretical physicist and 2017 Nobel laureate in Physics[4] Kip Thorne was
Cinematographer Hoyte van Hoytema shot it on 35 mm movie film in the Panavision ar
Principal photography began in late 2013 and took place in Alberta, Iceland, and I
Interstellar uses extensive practical and miniature effects and the company Double

Interstellar premiered on October 26, 2014, in Los Angeles.
In the United States, it was first released on film stock, expanding to venues usi
The film had a worldwide gross over $677 million (and $773 million with subsequent
It received acclaim for its performances, direction, screenplay, musical score, vi
It has also received praise from many astronomers for its scientific accuracy and
Interstellar was nominated for five awards at the 87th Academy Awards, winning Bes
```

Chunking:

In [ ]:

```
# Split into a list of sentences
texts = text.split('.')

# Clean up to remove empty spaces and new lines
texts = np.array([t.strip(' \n') for t in texts])
```

In [ ]:

```
texts
```

In [ ]:

```
# Split into a list of paragraphs
texts = text.split('\n\n')

# Clean up to remove empty spaces and new lines
texts = np.array([t.strip(' \n') for t in texts])
```

In [ ]:

```
texts
```

In [ ]:

```
# Split into a list of sentences
texts = text.split('.')

# Clean up to remove empty spaces and new lines
texts = np.array([t.strip(' \n') for t in texts])
```

In [ ]:

```
title = 'Interstellar (film)'

texts = np.array([f"{title} {t}" for t in texts])
```

In [ ]:

```
texts
```

Get the embeddings:

In [ ]:

```
response = co.embed(
    texts=texts.tolist()
).embeddings
```

In [ ]:

```
embeds = np.array(response)
embeds.shape
```

Create the search index:

In [ ]:

```
search_index = AnnoyIndex(embeds.shape[1], 'angular')
# Add all the vectors to the search index
for i in range(len(embeds)):
    search_index.add_item(i, embeds[i])

search_index.build(10) # 10 trees
search_index.save('test.ann')
```

In [ ]:

```
pd.set_option('display.max_colwidth', None)

def search(query):

  # Get the query's embedding
  query_embed = co.embed(texts=[query]).embeddings

  # Retrieve the nearest neighbors
  similar_item_ids = search_index.get_nns_by_vector(query_embed[0],
                                                    3,
                                                   include_distances=True)
  # Format the results
  results = pd.DataFrame(data={'texts': texts[similar_item_ids[0]],
                               'distance': similar_item_ids[1]})

  print(texts[similar_item_ids[0]])

  return results
```

In [ ]:

```
query = "How much did the film make?"
search(query)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: