Developing my own Language Model using the API of Cohere Embed LLM

I am using the API of the Embed LLM by Cohere, this language model is powered with RAG and semantic search that increases the accuracy of the model by using my own text data.

This model performs text generation from the data that it is trained below and analyzes text using emebddings. It has large number of sequence length and limited amount of attention heads.

It can only able to generate the output in the form of text based on prompt that is given and context window added here is small.

```
pip install cohere
```

```
Collecting cohere
  Downloading cohere-4.56-py3-none-any.whl (52 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52.7/52.7 kB 567.3 kB/s eta 0:00:00
Requirement already satisfied: aiohttp<4.0,>=3.0 in /usr/local/lib/python3.10/dist-packages (from cohere) (3.9.3)
Collecting backoff<3.0,>=2.0 (from cohere)
  Downloading backoff-2.2.1-py3-none-any.whl (15 kB)
Collecting fastavro<2.0,>=1.8 (from cohere)
  Downloading fastavro-1.9.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 10.1 MB/s eta 0:00:00
Collecting importlib_metadata<7.0,>=6.0 (from cohere)
  Downloading importlib_metadata-6.11.0-py3-none-any.whl (23 kB)
Requirement already satisfied: requests<3.0.0,>=2.25.0 in /usr/local/lib/python3.10/dist-packages (from cohere) (2.31.0)
Requirement already satisfied: urllib3<3,>=1.26 in /usr/local/lib/python3.10/dist-packages (from cohere) (2.0.7)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0,>=3.0->cohere) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0,>=3.0->cohere) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0,>=3.0->cohere) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0,>=3.0->cohere) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0,>=3.0->cohere) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp<4.0,>=3.0->cohere) (4.0.
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-packages (from importlib_metadata<7.0,>=6.0->cohere) (3.18.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.25.0->cohere
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.25.0->cohere) (3.6)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0.0,>=2.25.0->cohere) (202
Installing collected packages: importlib_metadata, fastavro, backoff, cohere
  Attempting uninstall: importlib_metadata
    Found existing installation: importlib_metadata 7.0.2
    Uninstalling importlib_metadata-7.0.2:
      Successfully uninstalled importlib_metadata-7.0.2
Successfully installed backoff-2.2.1 cohere-4.56 fastavro-1.9.4 importlib_metadata-6.11.0
```

```
pip install annoy
```

```
Collecting annoy
  Downloading annoy-1.17.3.tar.gz (647 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 647.5/647.5 kB 6.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: annoy
  Building wheel for annoy (setup.py) ... done
  Created wheel for annoy: filename=annoy-1.17.3-cp310-cp310-linux_x86_64.whl size=552450 sha256=7085aadefd705724d5880a562068e4806a36a0a
  Stored in directory: /root/.cache/pip/wheels/64/8a/da/f714bcf46c5efdcfcac0559e63370c21abe961c48e3992465a
Successfully built annoy
Installing collected packages: annoy
Successfully installed annoy-1.17.3
```

```
import cohere
import numpy as np
import warnings
warnings.filterwarnings('ignore')
from annoy import AnnoyIndex
import numpy as np
import pandas as pd
```

Training the Embed model with my text data that it require to store and retrieve at the time of text generation

```
#Sample prompt below
question = "Which Sydney beach should I visit?"


#Training data
text = """

Sydney is world famous for beautiful beaches. These beaches offer different vibes and attractions, from bustling crowds and great surfing co

Bondi Beach: Bondi is perhaps the most famous beach in Sydney, if not Australia. It's known for its golden sands, vibrant atmosphere, and ex

Manly Beach: Easily accessible by a scenic ferry ride from Circular Quay, Manly Beach is known for its laid-back atmosphere and family-frien

Cronulla Beach: Located in the Sutherland Shire, Cronulla offers a more relaxed atmosphere compared to some of the busier city beaches. It's

Bronte Beach: Situated between Bondi and Coogee, Bronte Beach is popular among both locals and visitors. It's a smaller, quieter beach with

Tamarama Beach: Also known as "Glamarama" due to its popularity among the fashion-conscious crowd, Tamarama Beach is a smaller and more secl

"""
```

Performing tokenization process to reduce the data into tiny form of chunks called as tokens and converting the tokens into the form of vectors using the process of Embedding. After conversion, it stores it in the vector database.

This helps for the model to easily understand the data in the form of vector spaces.

Note: This text to text conversion is powered with Embed model from Cohere. I am using my own data to train this LLM as the model is already integrated with RAG and semantic search for generating accurate responses.

```
# 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 if t])


# Using the API of Embed model from Cohere
co = cohere.Client('PbrfoQaDsYWDOIcqctLKJjKUakayNAJudibQafl8')


# Get the embeddings
response = co.embed(

    texts=texts.tolist(),

).embeddings
```

```
     default model on embed will be deprecated in the future, please specify a model in the request.
```

Performing indexing to perform semantic search for specifying embedding vectors at each index for the model to retreive the necessary result.

```
# Check the dimensions of the embeddings
embeds = np.array(response)

# Create the search index, pass the size of embedding
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')
```

```
     True
```

Collecting the embedding for to retreive certain similar information for generating output.

```
def search_text(query):

    # Get the query's embedding for the model to recognize the prompt and compare the prompt with the vectors in embedding space
    query_embed = co.embed(texts=[query]).embeddings



    # Retrieve the nearest neighbors for finding similiar texts
    similar_item_ids = search_index.get_nns_by_vector(query_embed[0],

                                                   10,

                                                 include_distances=True)



    search_results = texts[similar_item_ids[0]]



    return search_results

results = search_text(question)

print(results[0])
```

```
default model on embed will be deprecated in the future, please specify a model in the request.
Bondi Beach: Bondi is perhaps the most famous beach in Sydney, if not Australia. It's known for its golden sands, vibrant atmosphere, an
```

```
def ask_llm(question, num_generations=1):
results = ask_llm(question,)

print(results[0])

     default model on embed will be deprecated in the future, please specify a model in the request.
     Bondi Beach is recommended as a great beach to visit in Sydney with its golden sands, vibrant atmosphere and excellent surfing condition
```

Now the model is ready after giving it a text data and stored the vectors in the embedding database.

```
question = "Which Sydney beach is for family?"
results = ask_llm(question,)
print(results[0])


     you looking for a particular type of family-friendly environment?   It may also be useful to know what is considered a family-friendly b

    Question: {question}
question = "Sydney is considered as the family-friendly environment beaches, have you got it now?"
results = ask_llm(question,)
print(results[0])

     default model on embed will be deprecated in the future, please specify a model in the request.
     Sydney's beaches offer a variety of experiences, including family-friendly environments.


question = "Which Sydney beach has a rock pool?"
results = ask_llm(question,)
print(results[0])

     default model on embed will be deprecated in the future, please specify a model in the request.
     Bronte Beach is the Sydney beach with a rock pool.




        model="command-nightly",

        temperature=0.5,

        num_generations=num_generations

    )
    return prediction.generations
```