```
!pip install transformers==4.30 accelerate==0.22.0 einops==0.6.1 langchain==0.0.300 xformers==0.0.21 \
bitsandbytes==0.41.1 sentence_transformers==2.2.2 chromadb==0.4.12

!pip install accelerate
!pip install bitsandbytes

from google.colab import drive
drive.mount('/content/drive')

%cd /content/drive/My Drive/Learning Notebooks/
```

Data Prep

```
def read_text_file(file_path):
    try:
        with open(file_path, 'r',encoding='utf-8') as file:
            text = file.read()
        return text
    except FileNotFoundError:
        print("File not found!")
        return None
s2 = ''
for i in range(1977, 2001):
  file_content = read_text_file(f1+str(i)+".txt")
  s2+='. '+str(i)+ " year's letter below"
  s2+=file_content
s2

import locale
locale.getpreferredencoding = lambda: "UTF-8"
!pip install pypdf
from pypdf import PdfReader

def ret_text(s, n, fix):
  reader = PdfReader('BH/'+str(i)+fix+'.pdf')
  text = ''
  for j in range(len(reader.pages)):
    page = reader.pages[j]
    text += page.extract_text()
  t = '.  '+str(i)+" year's letter below"
  return t+' '+text
  # return str(i)
s=''
for i in [2001, 2002]:
  s+=ret_text(s, i, 'pdf')

for i in range(2003, 2024):
  s+=ret_text(s, i, 'ltr')
s

def save_string_to_txt(string_data, file_path):
    try:
        with open(file_path, 'w', encoding='utf-8') as file:
            file.write(string_data)
        print("String saved to", file_path)
    except Exception as e:
        print("Error:", e)

save_string_to_txt(s2+s, f2)
```

```
Requirement already satisfied: pypdf in /usr/local/lib/python3.10/dist-packages (4.2.0)
Requirement already satisfied: typing_extensions>=4.0 in /usr/local/lib/python3.10/dist-packages (from pypdf) (4.11.0)
String saved to BH_output_file.txt
```

RAG

```
from huggingface_hub import notebook_login

notebook_login()
```

```
%%time
from torch import cuda, bfloat16
import torch
import transformers
from transformers import AutoTokenizer
from langchain.llms import HuggingFacePipeline

from langchain.document_loaders import TextLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.chains import RetrievalQA
from langchain.vectorstores import Chroma

import accelerate
import bitsandbytes


model_id = "meta-llama/Llama-2-13b-chat-hf"

device = f'cuda:{cuda.current_device()}' if cuda.is_available() else 'cpu'

bnb_config = transformers.BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type='nf4',
    bnb_4bit_use_double_quant=True,
    bnb_4bit_compute_dtype=bfloat16
)
model_config = transformers.AutoConfig.from_pretrained(
    model_id,
)
model = transformers.AutoModelForCausalLM.from_pretrained(
    model_id,
    trust_remote_code=True,
    config=model_config,
    quantization_config=bnb_config,
    device_map='auto',
)

tokenizer = AutoTokenizer.from_pretrained(model_id)

query_pipeline = transformers.pipeline(
        "text-generation",
        model=model,
        tokenizer=tokenizer,
        torch_dtype=torch.float16,
        device_map="auto",
        # max_length=200
        max_new_tokens = 1000
        )

def test_model(tokenizer, pipeline, prompt_to_test):

    sequences = pipeline(
        prompt_to_test,
        do_sample=True,
        top_k=10,
        num_return_sequences=1,
        eos_token_id=tokenizer.eos_token_id,
```

```
        # max_length=200,
        max_new_tokens = 1000
        )

    for seq in sequences:
        print('Result: ')
        nthelement = 20
        items = seq['generated_text'].split(' ')
        groups = []

        while items:
            first_three, items = items[:nthelement], items[nthelement:]
            groups.append(first_three)

        result = "\n".join(" ".join(g) for g in groups)
        print(result)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens)
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100% 587/587 [00:00<00:00, 19.6kB/s]
model.safetensors.index.json: 100% 33.4k/33.4k [00:00<00:00, 1.99MB/s]
Downloading shards: 100% 3/3 [04:10<00:00, 78.43s/it]
model-00001-of-00003.safetensors: 100% 9.95G/9.95G [01:41<00:00, 94.3MB/s]
model-00002-of-00003.safetensors: 100% 9.90G/9.90G [01:30<00:00, 175MB/s]
model-00003-of-00003.safetensors: 100% 6.18G/6.18G [00:58<00:00, 150MB/s]
Loading checkpoint shards: 100% 3/3 [02:03<00:00, 38.89s/it]
generation_config.json: 100% 188/188 [00:00<00:00, 12.7kB/s]
tokenizer_config.json: 100% 1.62k/1.62k [00:00<00:00, 59.7kB/s]
tokenizer.model: 100% 500k/500k [00:00<00:00, 21.6MB/s]
tokenizer.json: 100% 1.84M/1.84M [00:00<00:00, 13.6MB/s]
special_tokens_map.json: 100% 414/414 [00:00<00:00, 29.7kB/s]
CPU times: user 53.5 s, sys: 1min 5s, total: 1min 59s
Wall time: 6min 39s
```

```
%%time
# testing
test_model(tokenizer,
           query_pipeline,
           "summarize what is the best way to measure any single year’s performance for any company")
```

```
Result:
summarize what is the best way to measure any single year’s performance for any company?
Answer: I like to use a
combination of metrics to evaluate a company’s performance in a single year. These can include:
• Net income: This measure the
company’s overall profitability for the year, taking into account all revenue and expenses.
```

```
• Revenue growth: This measures the increase in
revenue from the previous year and shows whether the company is growing and expanding its business.
• Gross margin: This measures
the profitability of the company's core operations, calculated as the difference between revenue and the cost of goods sold, expressed
as a percentage.
• Operating margin: This measures the profitability of the company's operations, calculated as the difference between gross profit
and operating expenses, expressed as a percentage.
• Return on equity (ROE): This measures the company's ability to generate profits from
shareholders' equity, calculated as net income divided by shareholders' equity.
• Return on assets (ROA): This measures the company's ability to
generate profits from its assets, calculated as net income divided by total assets.
• Cash flow from operations: This measures the
company's ability to generate cash from its operations, calculated as the cash provided by operating activities minus the cash used
by operating activities.

These metrics can give a comprehensive view of a company's performance in a single year, taking into account
both profitability and asset utilization. However, it is important to note that no single metric can tell the whole story,
and it is always best to consider a range of metrics when evaluating a company's performance.
CPU times: user 47.5 s, sys: 347 ms, total: 47.9 s
Wall time: 52.3 s
```

```
%%time

llm = HuggingFacePipeline(pipeline=query_pipeline)

llm(prompt="summarize what is the best way to measure any single year's performance for any company")
```

```
CPU times: user 1min 15s, sys: 0 ns, total: 1min 15s
Wall time: 1min 18s
'?\n\nAnswer: The best way to measure a single year's performance for any company is to use a combination of financial and non-financial metrics that provide a comprehensive view of
the company's performance. Here are some key metrics to consider:\n\n1. Revenue growth: Measure the year-over-year growth in revenue to assess the company's top-line performance.\n2.
Net income growth: Measure the year-over-year growth in net income to assess the company's profitability.\n3. Earnings per share (EPS): Measure the year-over-year growth in EPS to as
sess the company's profitability from a shareholder's perspective.\n4. Return on equity (ROE): Measure the company's ROE to assess its ability to generate profits from shareholders'
equity.\n5. Return on assets (ROA): Measure the company's ROA to assess its ability to generate profits from its assets.\n6. Gross margin: Measure the company's gross margin to asses
```

```
%%time

llm = HuggingFacePipeline(pipeline=query_pipeline)

llm(prompt="List out some instances where warren buffett emphasizes on patience as a vertue of an investor")
```

```
CPU times: user 1min 7s, sys: 141 ms, total: 1min 7s
Wall time: 1min 13s
'.\n\nWarren Buffett, one of the most successful investors in history, has consistently emphasized the importance of patience as a key virtue for investors. Here are some instances w
here he has stressed the importance of patience:\n\n1. In his 1994 letter to shareholders, Buffett wrote, "Our favorite holding period is forever." This quote highlights the importan
ce of patience in investing, as Buffett is willing to hold onto his investments for the long term, rather than trying to time the market or make quick profits.\n2. In a 2018 intervie
w with CNBC, Buffett said, "I think the most important thing is to have a long-term perspective. The more you think about the short term, the more you\'re going to get hurt." This qu
ote emphasizes the importance of patience in avoiding short-term thinking and focusing on the long-term potential of investments.\n3. In his 2017 letter to shareholders, Buffett wrot
e, "The weeds wither away, but the flowers remain." This quote is a metaphor for the importance of patience in investing, as the weeds (short-term challenges and setbacks) will event
ually wither away, but the flowers (long-term investments) will remain and flourish.\n4. In a 2016 interview with Bloomberg, Buffett said, "I\'ve never met a successful person who di
dn\'t have patience." This quote highlights the importance of patience in achieving success, whether in investing or in other areas of life.\n5. In his 2015 letter to shareholders, B
uffett wrote, "Patience is the key to success in investing." This quote is a direct statement of the importance of patience in investing, as Buffett believes that patience is the key
```

```
loader = TextLoader("BH_output_file.txt",
                    encoding="utf8")
documents = loader.load()
```

```
%%time
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=20)
all_splits = text_splitter.split_documents(documents)
```

```
model_name = "sentence-transformers/all-mpnet-base-v2"
model_kwargs = {"device": "cuda"}

embeddings = HuggingFaceEmbeddings(model_name=model_name, model_kwargs=model_kwargs)

vectordb = Chroma.from_documents(documents=all_splits, embedding=embeddings, persist_directory="chroma_db")


retriever = vectordb.as_retriever()

qa = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=retriever,
    verbose=True
)

def test_rag(qa, query):
    print(f"Query: {query}\n")
    result = qa.run(query)
    print("\nResult: ", result)
```

```
.gitattributes: 100% 1.23k/1.23k [00:00<00:00, 69.2kB/s]
1_Pooling/config.json: 100% 190/190 [00:00<00:00, 10.0kB/s]
README.md: 100% 10.6k/10.6k [00:00<00:00, 823kB/s]
config.json: 100% 571/571 [00:00<00:00, 35.4kB/s]
config_sentence_transformers.json: 100% 116/116 [00:00<00:00, 8.71kB/s]
data_config.json: 100% 39.3k/39.3k [00:00<00:00, 1.74MB/s]
model.safetensors: 100% 438M/438M [00:02<00:00, 154MB/s]
pytorch_model.bin: 100% 438M/438M [00:02<00:00, 168MB/s]
sentence_bert_config.json: 100% 53.0/53.0 [00:00<00:00, 3.59kB/s]
special_tokens_map.json: 100% 239/239 [00:00<00:00, 16.4kB/s]
tokenizer.json: 100% 466k/466k [00:00<00:00, 25.4MB/s]
tokenizer_config.json: 100% 363/363 [00:00<00:00, 15.4kB/s]
train_script.py: 100% 13.1k/13.1k [00:00<00:00, 569kB/s]
vocab.txt: 100% 232k/232k [00:00<00:00, 12.3MB/s]
modules.json: 100% 349/349 [00:00<00:00, 25.7kB/s]
CPU times: user 1min 35s, sys: 1.62 s, total: 1min 37s
Wall time: 1min 48s
```

```
%%time

query = "summarize what is the best way to measure any single year’s performance for any company"
test_rag(qa, query)
```

```
Query: summarize what is the best way to measure any single year’s performance for any company
```

```
> Entering new RetrievalQA chain...

> Finished chain.

Result:

The best way to measure any single year's performance for any company is to use the change in the company's per-share intrinsic value over that year, rather than the change in its stc
CPU times: user 38 s, sys: 36.1 ms, total: 38.1 s
Wall time: 38.4 s
```

```
%%time
# doc sources

docs = vectordb.similarity_search(query)
print(f"Query: {query}")
print(f"Retrieved documents: {len(docs)}")
for doc in docs:
    doc_details = doc.to_json()['kwargs']
    print("Source: ", doc_details['metadata']['source'])
    print("Text: ", doc_details['page_content'], "\n")
```

```
Query: summarize what is the best way to measure any single year’s performance for any company
Retrieved documents: 4
Source:  BH_output_file.txt
Text:  good arguments for simply using the change in our stock price. Over an extended period of time, in fact, that isthe best test. But year-to-year market prices can be extraordina
The ideal standard for measuring our yearly progress would be the change in Berkshire’s per-share intrinsic

Source:  BH_output_file.txt
Text:  Long Term Results

     In measuring long term economic performance - in contrast to
yearly performance - we believe it is appropriate to recognize
fully any realized capital gains or losses as well as
extraordinary items, and also to utilize financial statements
presenting equity securities at market value.  Such capital gains
or losses, either realized or unrealized, are fully as important
to shareholders over a period of years as earnings realized in a
more routine manner through operations; it is just that their
impact is often extremely capricious in the short run, a
characteristic that makes them inappropriate as an indicator of
single year managerial performance.

Source:  BH_output_file.txt
Text:  You should keep at least three points in mind as you
evaluate this data.  The first point concerns the many businesses
we operate whose annual earnings are unaffected by changes in
stock market valuations.  The impact of these businesses on both
```