LLM(Llama v2) with pdf context

In previous 2 notebooks we have individually loaded Llama 7b and FAISS vectorstore . In this we combine both for incontext Q and A

Installing required libraries

1. transformers: Hugging face libraries for loading pretrained transformer checkpoints.
2. accelerate : Manages communications between CPU and GPU more efficiently.
3. datasets : For loading datasets from huggingface for future fine tuning.
4. bitsandbytes : Used for quantization of model to run on limited resources.
5. einops :Einstein-Inspired Notation for operations
6. wandb :weights and biases for visualizations
7. Langchain : High level library assisting in creation and deployment of LLM apps : https://python.langchain.com/docs/get_started
8. sentence_transformers : SentenceTransformers 🤗 is a Python framework for state-of-the-art sentence, text and image embeddings. : https://huggingface.co/sentence-transformers
9. faiss-cpu / faiss-gpu :Facebook AI Similarity Search (Faiss) is a library for efficient similarity search and clustering of dense vectors. : https://python.langchain.com/docs/integrations/vectorstores/faiss
10. pypdf : pypdf into array of documents, where each document contains the page content and metadata with page number.: https://python.langchain.com/docs/modules/data_connection/document_loaders/pdf
11. tiktoken : Used for tokenwise splitting of string or steam . Tbn that tokenization takes place according to tokens not words.

```
!pip install -q -U trl transformers accelerate
!pip install -q datasets bitsandbytes einops wandb
!pip install langchain
!pip install sentence_transformers
# !pip install faiss-cpu
!pip install faiss-gpu
!pip install pypdf
!pip install  tiktoken
```

```
  Building wheels for collected packages: sentence_transformers
    Building wheel for sentence_transformers (setup.py) ... done
    Created wheel for sentence_transformers: filename=sentence_transformers-2.2.2-py3-none-any.whl size=125925
    Stored in directory: /root/.cache/pip/wheels/62/f2/10/1e606fd5f02395388f74e7462910fe851042f97238cbbd902f
  Successfully built sentence_transformers
  Installing collected packages: sentencepiece, sentence_transformers
  Successfully installed sentence_transformers-2.2.2 sentencepiece-0.1.99
  Collecting faiss-gpu
    Downloading faiss_gpu-1.7.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (85.5 MB)
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.5/85.5 MB 9.3 MB/s eta 0:00:00
  Installing collected packages: faiss-gpu
  Successfully installed faiss-gpu-1.7.2
  Collecting pypdf
    Downloading pypdf-3.14.0-py3-none-any.whl (269 kB)
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 269.8/269.8 kB 4.7 MB/s eta 0:00:00
  Installing collected packages: pypdf
  Successfully installed pypdf-3.14.0
  Collecting tiktoken
    Downloading tiktoken-0.4.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.7 MB)
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 9.5 MB/s eta 0:00:00
  Requirement already satisfied: regex>=2022.1.18 in /usr/local/lib/python3.10/dist-packages (from tiktoken) (2
  Requirement already satisfied: requests>=2.26.0 in /usr/local/lib/python3.10/dist-packages (from tiktoken) (2
  Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from request
  Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=
  Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from req
  Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.26.0
  Installing collected packages: tiktoken
```

```
from huggingface_hub import login
login()
```

```
Token is valid (permission: read).
ı has been saved in your configured git credential helpe
ır token has been saved to /root/.cache/huggingface/tok
Login successful
```

```
from transformers import AutoModelForCausalLM, AutoTokenizer, BitsAndBytesConfig
from transformers import pipeline
import transformers
import torch
```

pdftextloader : https://python.langchain.com/docs/modules/data_connection/document_loaders/pdf

```
from langchain.embeddings import HuggingFaceEmbeddings
from langchain.text_splitter import CharacterTextSplitter
from langchain.vectorstores import FAISS
from langchain.document_loaders import TextLoader
from langchain.document_loaders import PyPDFLoader
from langchain.document_loaders import UnstructuredURLLoader
from langchain.document_loaders import UnstructuredPDFLoader
from langchain.document_loaders import PDFMinerPDFasHTMLLoader
from langchain.document_loaders import PDFPlumberLoader
from langchain.text_splitter import TokenTextSplitter

from IPython.display import HTML

from langchain.document_loaders import OnlinePDFLoader
```

```
model_name = "meta-llama/Llama-2-7b-chat-hf"

bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
```

```
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.float16,
)

model = AutoModelForCausalLM.from_pretrained(
    model_name,
    quantization_config=bnb_config,
    trust_remote_code=True
)
model.config.use_cache = False

tokenizer = AutoTokenizer.from_pretrained(model_name, trust_remote_code=True)
tokenizer.pad_token = tokenizer.eos_token
```

```
Downloading (…)lve/main/config.json: 100% 614/614 [00:00<00:00, 18.8kB/s]
Downloading (…)fetensors.index.json: 100% 26.8k/26.8k [00:00<00:00, 1.18MB/s]
Downloading shards: 100% 2/2 [07:16<00:00, 189.15s/it]
Downloading (…)of-00002.safetensors: 100% 9.98G/9.98G [06:22<00:00, 22.7MB/s]
Downloading (…)of-00002.safetensors: 100% 3.50G/3.50G [00:53<00:00, 28.1MB/s]
Loading checkpoint shards: 100% 2/2 [01:12<00:00, 32.99s/it]
```

```
from IPython.display import Markdown
if torch.cuda.is_available():
    print('cuda')
    device = torch.device("cuda")  # If GPU is available, use it.
else:
    device = torch.device("cpu")   # If GPU is not available, use the CPU.
```

```
cuda
```

Observations :

1. Sometimes the 404 Error occurs where the link to the urls is not accessible even if url is correct . In such case restarting the runtime helped.
2. I observed that formating of pypdf was the best . but the rest of the loaders could also be used. PDF loaders : https://python.langchain.com/docs/modules/data_connection/document_loaders/pdf

Part 1 : Going Sequentially

Loading Model --> Loading Embeddings and Vector Store --> Loading Context --> Splitting ,embedding and saving context --> Passing instruction and finding nearest context --> Creating Pormpt --> Passing prompt to LLM.

```
urls="https://arxiv.org/pdf/1706.03762.pdf"
loader=PyPDFLoader(urls)
data=loader.load()
```

```
content=' '
for i in data : content+=i.page_content
content
```

```
' Provided proper attribution is provided, Google hereby grants permission to\nre
produce the tables and figures in this paper solely for use in journalistic or\ns
cholarly works.\nAttention Is All You Need\nAshish Vaswani∗\nGoogle Brain\navaswa
ni@google.comNoam Shazeer∗\nGoogle Brain\nnoam@google.comNiki Parmar∗\nGoogle Res
earch\nnikip@google.comJakob Uszkoreit∗\nGoogle Research\nusz@google.com\nLlion J
ones∗\nGoogle Research\nllion@google.comAidan N. Gomez∗ †\nUniversity of Toronto
```

```
text_splitter = TokenTextSplitter(chunk_size=1000, chunk_overlap=200)

texts = text_splitter.create_documents([content])
```

```
embeddings = HuggingFaceEmbeddings()
```

```
Downloading (…)a8e1d/.gitattributes: 100% 1.18k/1.18k [00:00<00:00, 75.4kB/s]
Downloading (…)_Pooling/config.json: 100% 190/190 [00:00<00:00, 14.7kB/s]
Downloading (…)b20bca8e1d/README.md: 100% 10.6k/10.6k [00:00<00:00, 546kB/s]
Downloading (…)0bca8e1d/config.json: 100% 571/571 [00:00<00:00, 40.5kB/s]
Downloading (…)ce_transformers.json: 100% 116/116 [00:00<00:00, 7.74kB/s]
Downloading (…)e1d/data_config.json: 100% 39.3k/39.3k [00:00<00:00, 2.50MB/s]
Downloading pytorch_model.bin: 100% 438M/438M [00:02<00:00, 217MB/s]
Downloading 53.0/53.0 [00:00<00:00,
```

```
db = FAISS.from_documents(texts, embeddings)
```

```
question="What are the parts of transformers?"
contexts=db.similarity_search(question,k=2)
print(contexts)
```

```
[Document(page_content='former follows this overall architecture using stacked self-attention and point-wise, f
```

```
context=' '
for i in contexts:context+=i.page_content
context
```

```
' former follows this overall architecture using stacked self-attention and point
-wise, fully\nconnected layers for both the encoder and decoder, shown in the lef
t and right halves of Figure 1,\nrespectively.\n3.1 Encoder and Decoder Stacks\nE
ncoder: The encoder is composed of a stack of N= 6 identical layers. Each layer h
as two\nsub-layers. The first is a multi-head self-attention mechanism, and the s
econd is a simple, position-\nwise fully connected feed-forward network. We emplo
```

Input prompt for LLama is of the form :

```
prompt="""<s> [INST]
<<SYS>>
System guiding message
```

```
<</SYS>>
[/INST]"""
```

Sources:

1. https://huggingface.co/blog/llama2
2. https://discuss.huggingface.co/t/llama-2-7b-hf-repeats-context-of-question-directly-from-input-prompt-cuts-off-with-newlines/48250/10
   If not done so will result in the language model misbehaving . Therefore prompt engineering is important.

```
prompt=f"""<s>[INST] <<SYS>>
You are an honest assistant , who gives factually correct answers while refering to the context. If the context does
Answer the following question while refering the context.If the answer is not in context ,try to answer on your own
<</SYS>>
Context:{context}
Qustion:{question}
 [/INST]
"""
print(prompt)
```

```
The code we used to train and evaluate our models is available at https://github.com/
tensorflow/t
Qustion:What are the parts of transformers?
 [/INST]
```

```
from IPython.display import Markdown
```

```
# prompt = 'I liked "Breaking Bad" and "Band of Brothers". Do you have any recommendations of other shows I might li
inputs = tokenizer(prompt, return_tensors="pt",padding =True).to(device)
generate_ids = model.generate(inputs.input_ids, max_length=5000,top_k=1,top_p=0.5,temperature=0)
```

```
---------------------------------------------------------------------------
OutOfMemoryError                          Traceback (most recent call last)
<ipython-input-34-8a8e2b045d92> in <cell line: 3>()
      1 # prompt = 'I liked "Breaking Bad" and "Band of Brothers". Do you have
any recommendations of other shows I might like?'
      2 inputs = tokenizer(prompt, return_tensors="pt",padding =True).to(device)
----> 3 generate_ids = model.generate(inputs.input_ids,
max_length=5000,top_k=1,top_p=0.5,temperature=0)

                                 14 frames
/usr/local/lib/python3.10/dist-
packages/transformers/models/llama/modeling_llama.py in forward(self,
hidden_states, attention_mask, position_ids, past_key_value, output_attentions,
use_cache)
    328         value_states = repeat_kv(value_states, self.num_key_value_groups)
    329
--> 330         attn_weights = torch.matmul(query_states, key_states.transpose(2,
3)) / math.sqrt(self.head_dim)
    331
    332         if attn_weights.size() != (bsz, self.num_heads, q_len,
kv_seq_len):
```

Skipping the prompt from the result.

```
p=generate_ids[0][inputs['input_ids'].shape[1]:]
```

Markdown for better outputs. Results are quite astonishing for a 7b parameter model.

```
Markdown(tokenizer.decode(p, skip_special_tokens=True, clean_up_tokenization_spaces=True))
# print(tokenizer.batch_decode(generate_ids, skip_special_tokens=True, clean_up_tokenization_spaces=False)[0])
```

The Transformer architecture consists of the following parts:

1. Encoder: The encoder is composed of a stack of N identical layers, each of which consists of two sub-layers: a multi-head self-attention mechanism and a position-wise feed-forward network. The output of each sub-layer is passed through a residual connection and layer normalization.
2. Decoder: The decoder is also composed of a stack of N identical layers, with

Part 2 : Creating Chain using Langchain

input values, where the weights are computed based on the similarity between

```
from langchain.chains import LLMChain
from langchain.prompts import PromptTemplate
```

consists of multiple attention layers running in parallel, each with its own set of

```
prompt="""<s>[INST] <<SYS>>
You are an honest assistant , who gives factually correct answers while refering to the context. If the context does
Answer the following question while refering the context.If the answer is not in context ,try to answer on your own
<</SYS>>
Context:{context}
Qustion:{question}
 [/INST]
"""
```

3. Embeddings: Embeddings are used to represent each input value in a higher-dimensional space. They are typically learned during training and are used to

```
prompt_t=PromptTemplate(input_variables=['context','question'],template=prompt)
prompt_t
```

```
PromptTemplate(input_variables=['context', 'question'], output_parser=None, partial_variables={},
template='<s>[INST] <<SYS>>\nYou are an honest assistant , who gives factually correct answers while refering
to the context. If the context does not have an answer you try your best to answer the question on your
own.\n\nAnswer the following question while refering the context.If the answer is not in context ,try to
answer on your own . Still if you are not able to answer the question then politely say so instead of
rambling.\n\n<</SYS>>\nContext:{context}\nQustion:{question}\n [/INST]\n', template_format='f-string',
validate_template=True)
```

```
LLMChain?
```

```
chain=LLMChain(llm=model
               ,prompt=prompt_t)
```

```
---------------------------------------------------------------------------
ValidationError                           Traceback (most recent call last)
<ipython-input-36-4441de536e63> in <cell line: 1>()
----> 1 chain=LLMChain(llm=model
      2                ,prompt=prompt_t)

                                 1 frames
/usr/local/lib/python3.10/dist-packages/pydantic/main.cpython-310-x86_64-linux-
gnu.so in pydantic.main.BaseModel.__init__()

ValidationError: 1 validation error for LLMChain
llm
  value is not a valid dict (type=type_error.dict)

SEARCH STACK OVERFLOW
```

```
class Cust_Chain_obj():
  def __init__(self,model,FAISS_obj,context='',embeddings = HuggingFaceEmbeddings(),text_splitter = TokenTextSplitte
    self.FAISS_obj=FAISS_obj
    self.model=model
```

0s completed at 10:28 PM