Chat with Multiple PDFs using Llama 2 and LangChain

Can you build a chatbot that can answer questions from multiple PDFs? Can you do it with a private LLM? In this tutorial, we'll use the latest Llama 2 13B GPTQ model to chat with multiple PDFs. We'll use the LangChain library to create a chain that can retrieve relevant documents and answer questions from them.

You'll learn how to load a GPTQ model using AutoGPTQ, convert a directory with PDFs to a vector store and create a chain using LangChain that works with text chunks from the vector store.

In this part, we will be using Jupyter Notebook to run the code. If you prefer to follow along, you can find the notebook on GitHub: GitHub Repository

Llama 2

Llama 2[1] is the latest LLM offering from Meta AI! This cutting-edge language model comes with an expanded context window of *4096* tokens and an impressive *2T* token dataset, surpassing its predecessor, Llama 1, in various aspects. The best part? Llama 2 is free for commercial use (with restrictions). Packed with pre-trained and fine-tuned LLMs ranging from 7 billion to 70 billion parameters, these models are set to outperform existing open-source chat models on a wide range of benchmarks. Here's an overview of the models available in Llama 2:

Model	Llama2	Llama2-hf	Llama2-chat	Llama2-chat-hf
7B	Link	Link	Link	Link
13B	Link	Link	Link	Link
70B	Link	Link	Link	Link

Llama 2 comes in two primary versions - the base model and Llama-2-Chat - optimized for dialogue use cases.

But how good is Llama 2? Looking at the HuggingFace Open LLM Leaderboard[2], looks like Llama 2 (and modified versions of it) takes the top spots.

⚠️ While many sources claim that Llama 2 is Open Source, the license is not considered Open Source by the Open Source Definition[3]. You can read more about it here: https://blog.opensource.org/metas-llama-2-license-is-not-open-source/

GPTQ

GPTQ[4] is a post-training quantization method capable of efficiently compressing models with hundreds of billions of parameters to just 3 or 4 bits per parameter, with minimal loss of accuracy. The method's efficiency is evident by its ability to quantize large models like OPT-175B and BLOOM-176B in about four GPU hours, maintaining a high level of accuracy.

Moreover, the original paper demonstrates the method's robustness even in extreme quantization scenarios, where models are quantized to just 2 bits per component. It also includes practical implementations, running the compressed OPT-175B model efficiently on a single NVIDIA A100 GPU or a few NVIDIA A6000 GPUs, achieving impressive speedups of around 3.25x and 4.5x, respectively.

Here is a summary of GPTQ using LLaMa (from the GitHub[5] repository):

Wiki2 PPL	FP16	4bit-RTN	4bit-GPTQ	3bit-RTN	3bit-GPTQ	3g128-GPTQ
LLaMa-7B	5.68	6.29	**6.09**	25.54	**8.07**	6.61
LLaMa-13B	5.09	5.53	**5.36**	11.40	**6.63**	5.62
LLaMa-30B	4.10	4.54	**4.45**	14.89	**5.69**	4.80

Wiki2 PPL	FP16	4bit-RTN	4bit-GPTQ	3bit-RTN	3bit-GPTQ	3g128-GPTQ
LLaMa-65B	3.53	3.92	**3.84**	10.59	**5.04**	4.17

The table shows the perplexity of the Llama 1 models on the WikiText-2 dataset. Note that GPTQ is performing better than RTN (Round To Nearest) and is close to FP16.

Creating and Running GPTQ Models

In this tutorial, we'll use a GPTQ version of the Llama 2 13B chat model to chat with multiple PDFs. We'll use the `TheBloke/Llama-2-13B-chat-GPTQ` model from the HuggingFace model hub.

The models available in the repository were created using AutoGPTQ[6]. The library allows you to apply the GPTQ algorithm to a model and quantize it to 3 or 4 bits. The library also provides a `AutoGPTQForCausalLM` class that can be used to load the quantized model and do inference using it.

Setup

Let's start by installing the required libraries and downloading the model:

```
!pip install -Uqqq pip --progress-bar off
!pip install -qqq torch==2.0.1 --progress-bar off
!pip install -qqq transformers==4.31.0 --progress-bar off
!pip install -qqq langchain==0.0.266 --progress-bar off
!pip install -qqq chromadb==0.4.5 --progress-bar off
!pip install -qqq pypdf==3.15.0 --progress-bar off
!pip install -qqq xformers==0.0.20 --progress-bar off
!pip install -qqq sentence_transformers==2.2.2 --progress-bar off
!pip install -qqq InstructorEmbedding==1.0.1 --progress-bar off
!pip install -qqq pdf2image==1.16.3 --progress-bar off
```

The AutoGPTQ library is a bit special (since we need to install the correct version with CUDA support). We can download and install it using the following commands:

```
!wget -q https://github.com/PanQiWei/AutoGPTQ/releases/download/v0.4.0/auto_g
!pip install -qqq auto_gptq-0.4.0+cu118-cp310-cp310-linux_x86_64.whl --progre
```

Finally, we need to install the `poppler-utils` library to convert PDFs to images:

```
!sudo apt-get install poppler-utils
```

Now, let's import the required libraries:

```python
import torch
from auto_gptq import AutoGPTQForCausalLM
from langchain import HuggingFacePipeline, PromptTemplate
from langchain.chains import RetrievalQA
from langchain.document_loaders import PyPDFDirectoryLoader
from langchain.embeddings import HuggingFaceInstructEmbeddings
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.vectorstores import Chroma
from pdf2image import convert_from_path
from transformers import AutoTokenizer, TextStreamer, pipeline

DEVICE = "cuda:0" if torch.cuda.is_available() else "cpu"
```

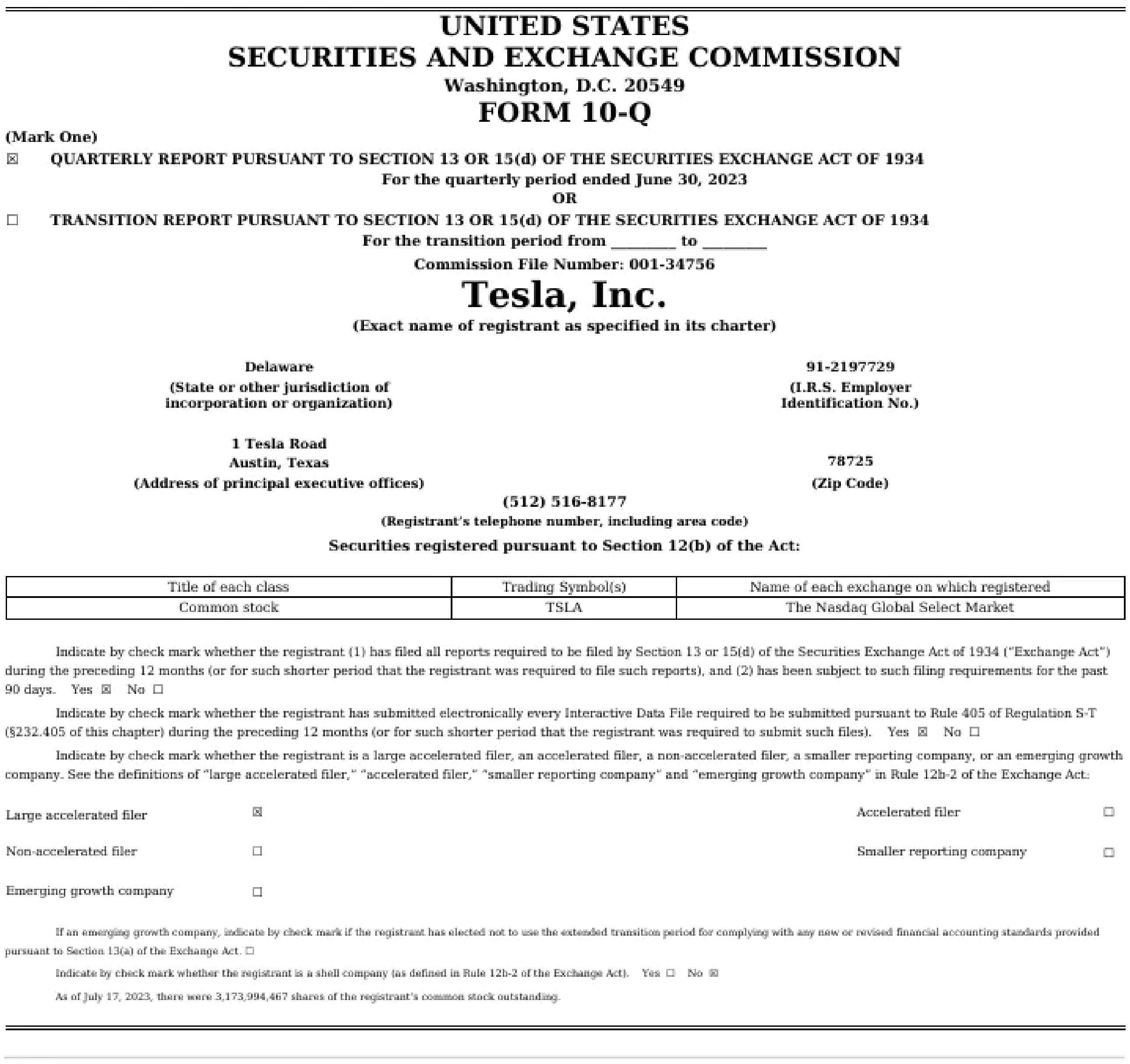
Data

We'll work with three PDFs in this tutorial:

```
!mkdir pdfs
!gdown 1v-Rn1FVU1pLTAQEgm0N9oB6cExMoebZr -O pdfs/tesla-earnings-report.pdf
!gdown 1Xc890jrQvCExAkryVWAttsv1DBLdVefN -O pdfs/nvidia-earnings-report.pdf
!gdown 1Epz-SQ3idPpoz75GlTzzomag8gplzLv8 -O pdfs/meta-earnings-report.pdf
```

These are the latest earning reports from Tesla, Nvidia, and Meta (formerly Facebook). We'll use these PDFs to chat with the Llama 2 13B GPTQ model.

Here's a preview of the first page of each PDF:



UNITED STATES
SECURITIES AND EXCHANGE COMMISSION
Washington, D.C. 20549
FORM 10-Q

(Mark One)

☒ **QUARTERLY REPORT PURSUANT TO SECTION 13 OR 15(d) OF THE SECURITIES EXCHANGE ACT OF 1934**
For the quarterly period ended June 30, 2023
OR
☐ **TRANSITION REPORT PURSUANT TO SECTION 13 OR 15(d) OF THE SECURITIES EXCHANGE ACT OF 1934**
For the transition period from ________ to ________
Commission File Number: 001-34756

Tesla, Inc.
(Exact name of registrant as specified in its charter)

Delaware	**91-2197729**
(State or other jurisdiction of incorporation or organization)	**(I.R.S. Employer Identification No.)**
1 Tesla Road Austin, Texas	**78725**
(Address of principal executive offices)	**(Zip Code)**

(512) 516-8177
(Registrant's telephone number, including area code)

Securities registered pursuant to Section 12(b) of the Act:

Title of each class	Trading Symbol(s)	Name of each exchange on which registered
Common stock	TSLA	The Nasdaq Global Select Market

Indicate by check mark whether the registrant (1) has filed all reports required to be filed by Section 13 or 15(d) of the Securities Exchange Act of 1934 ("Exchange Act") during the preceding 12 months (or for such shorter period that the registrant was required to file such reports), and (2) has been subject to such filing requirements for the past 90 days. Yes ☒ No ☐

Indicate by check mark whether the registrant has submitted electronically every Interactive Data File required to be submitted pursuant to Rule 405 of Regulation S-T (§232.405 of this chapter) during the preceding 12 months (or for such shorter period that the registrant was required to submit such files). Yes ☒ No ☐

Indicate by check mark whether the registrant is a large accelerated filer, an accelerated filer, a non-accelerated filer, a smaller reporting company, or an emerging growth company. See the definitions of "large accelerated filer," "accelerated filer," "smaller reporting company" and "emerging growth company" in Rule 12b-2 of the Exchange Act:

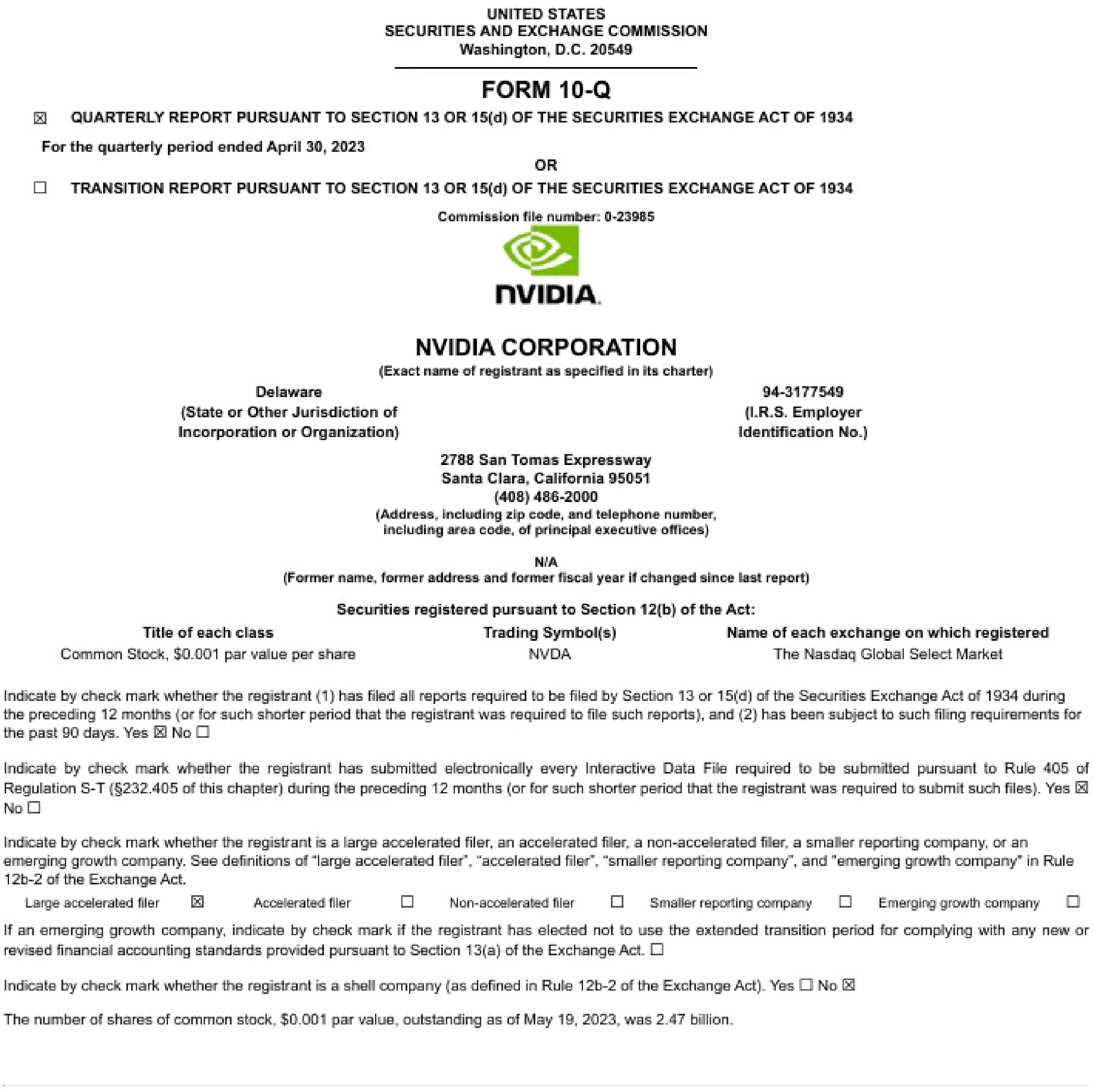
Large accelerated filer	☒	Accelerated filer	☐
Non-accelerated filer	☐	Smaller reporting company	☐
Emerging growth company	☐		

If an emerging growth company, indicate by check mark if the registrant has elected not to use the extended transition period for complying with any new or revised financial accounting standards provided pursuant to Section 13(a) of the Exchange Act. ☐

Indicate by check mark whether the registrant is a shell company (as defined in Rule 12b-2 of the Exchange Act). Yes ☐ No ☒

As of July 17, 2023, there were 3,173,994,467 shares of the registrant's common stock outstanding.

Tesla Earnings Sample Page



UNITED STATES
SECURITIES AND EXCHANGE COMMISSION
Washington, D.C. 20549

FORM 10-Q

☒ QUARTERLY REPORT PURSUANT TO SECTION 13 OR 15(d) OF THE SECURITIES EXCHANGE ACT OF 1934

For the quarterly period ended April 30, 2023

OR

☐ TRANSITION REPORT PURSUANT TO SECTION 13 OR 15(d) OF THE SECURITIES EXCHANGE ACT OF 1934

Commission file number: 0-23985

NVIDIA.

NVIDIA CORPORATION
(Exact name of registrant as specified in its charter)

Delaware	94-3177549
(State or Other Jurisdiction of Incorporation or Organization)	(I.R.S. Employer Identification No.)

2788 San Tomas Expressway
Santa Clara, California 95051
(408) 486-2000
(Address, including zip code, and telephone number, including area code, of principal executive offices)

N/A
(Former name, former address and former fiscal year if changed since last report)

Securities registered pursuant to Section 12(b) of the Act:

Title of each class	Trading Symbol(s)	Name of each exchange on which registered
Common Stock, $0.001 par value per share	NVDA	The Nasdaq Global Select Market

Indicate by check mark whether the registrant (1) has filed all reports required to be filed by Section 13 or 15(d) of the Securities Exchange Act of 1934 during the preceding 12 months (or for such shorter period that the registrant was required to file such reports), and (2) has been subject to such filing requirements for the past 90 days. Yes ☒ No ☐

Indicate by check mark whether the registrant has submitted electronically every Interactive Data File required to be submitted pursuant to Rule 405 of Regulation S-T (§232.405 of this chapter) during the preceding 12 months (or for such shorter period that the registrant was required to submit such files). Yes ☒ No ☐

Indicate by check mark whether the registrant is a large accelerated filer, an accelerated filer, a non-accelerated filer, a smaller reporting company, or an emerging growth company. See definitions of "large accelerated filer", "accelerated filer", "smaller reporting company", and "emerging growth company" in Rule 12b-2 of the Exchange Act.

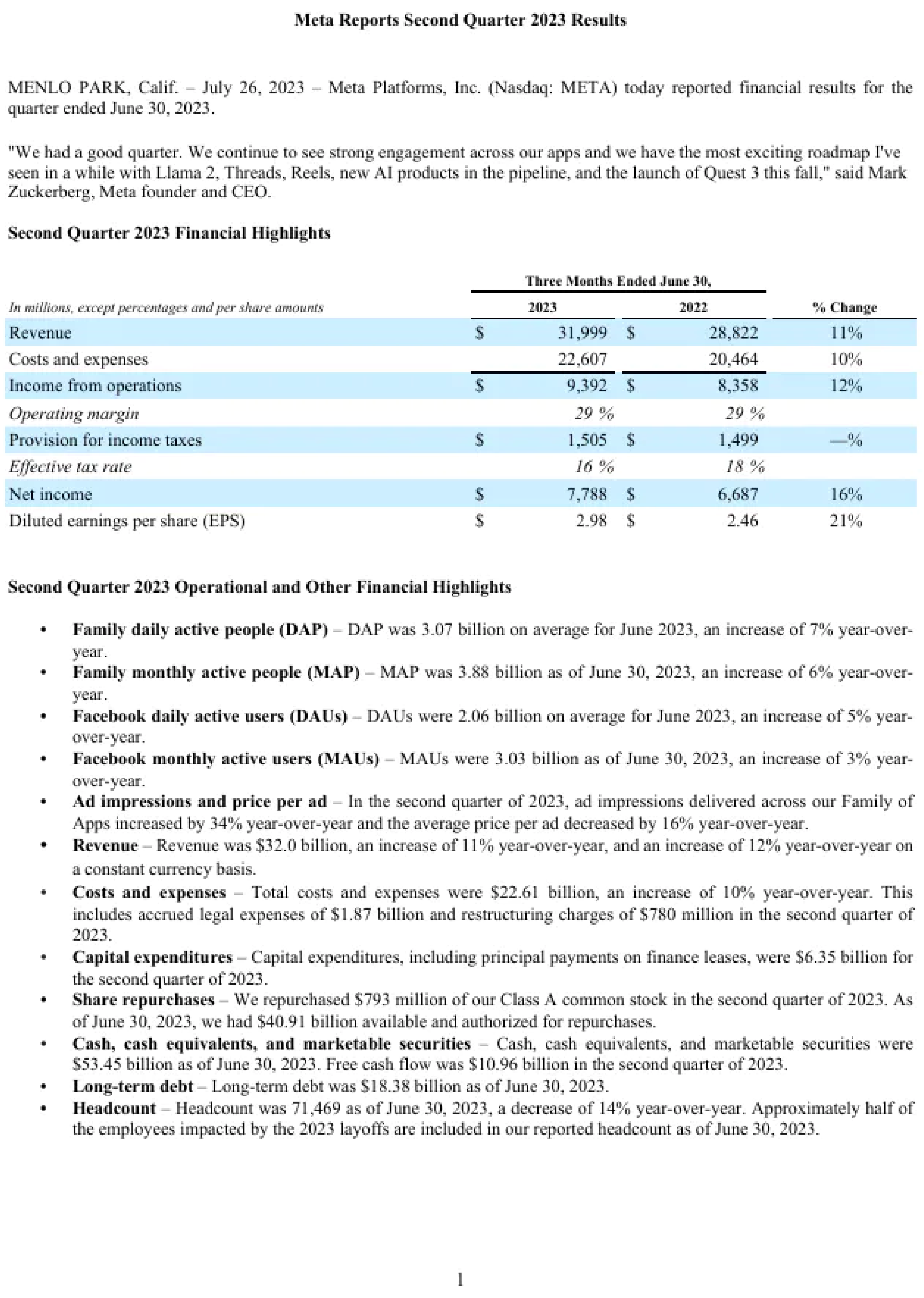
Large accelerated filer	☒	Accelerated filer	☐	Non-accelerated filer	☐	Smaller reporting company	☐	Emerging growth company	☐

If an emerging growth company, indicate by check mark if the registrant has elected not to use the extended transition period for complying with any new or revised financial accounting standards provided pursuant to Section 13(a) of the Exchange Act. ☐

Indicate by check mark whether the registrant is a shell company (as defined in Rule 12b-2 of the Exchange Act). Yes ☐ No ☒

The number of shares of common stock, $0.001 par value, outstanding as of May 19, 2023, was 2.47 billion.

Nvidia Earnings Sample Page



Meta Reports Second Quarter 2023 Results

MENLO PARK, Calif. – July 26, 2023 – Meta Platforms, Inc. (Nasdaq: META) today reported financial results for the quarter ended June 30, 2023.

"We had a good quarter. We continue to see strong engagement across our apps and we have the most exciting roadmap I've seen in a while with Llama 2, Threads, Reels, new AI products in the pipeline, and the launch of Quest 3 this fall," said Mark Zuckerberg, Meta founder and CEO.

Second Quarter 2023 Financial Highlights

In millions, except percentages and per share amounts	Three Months Ended June 30, 2023	2022	% Change
Revenue	$ 31,999	$ 28,822	11%
Costs and expenses	22,607	20,464	10%
Income from operations	$ 9,392	$ 8,358	12%
Operating margin	*29 %*	*29 %*	
Provision for income taxes	$ 1,505	$ 1,499	—%
Effective tax rate	*16 %*	*18 %*	
Net income	$ 7,788	$ 6,687	16%
Diluted earnings per share (EPS)	$ 2.98	$ 2.46	21%

Second Quarter 2023 Operational and Other Financial Highlights

- **Family daily active people (DAP)** – DAP was 3.07 billion on average for June 2023, an increase of 7% year-over-year.
- **Family monthly active people (MAP)** – MAP was 3.88 billion as of June 30, 2023, an increase of 6% year-over-year.
- **Facebook daily active users (DAUs)** – DAUs were 2.06 billion on average for June 2023, an increase of 5% year-over-year.
- **Facebook monthly active users (MAUs)** – MAUs were 3.03 billion as of June 30, 2023, an increase of 3% year-over-year.
- **Ad impressions and price per ad** – In the second quarter of 2023, ad impressions delivered across our Family of Apps increased by 34% year-over-year and the average price per ad decreased by 16% year-over-year.
- **Revenue** – Revenue was $32.0 billion, an increase of 11% year-over-year, and an increase of 12% year-over-year on a constant currency basis.
- **Costs and expenses** – Total costs and expenses were $22.61 billion, an increase of 10% year-over-year. This includes accrued legal expenses of $1.87 billion and restructuring charges of $780 million in the second quarter of 2023.
- **Capital expenditures** – Capital expenditures, including principal payments on finance leases, were $6.35 billion for the second quarter of 2023.
- **Share repurchases** – We repurchased $793 million of our Class A common stock in the second quarter of 2023. As of June 30, 2023, we had $40.91 billion available and authorized for repurchases.
- **Cash, cash equivalents, and marketable securities** – Cash, cash equivalents, and marketable securities were $53.45 billion as of June 30, 2023. Free cash flow was $10.96 billion in the second quarter of 2023.
- **Long-term debt** – Long-term debt was $18.38 billion as of June 30, 2023.
- **Headcount** – Headcount was 71,469 as of June 30, 2023, a decrease of 14% year-over-year. Approximately half of the employees impacted by the 2023 layoffs are included in our reported headcount as of June 30, 2023.

1

Meta Earnings Sample Page

These documents are long and complex. They use technical language and contain a lot of numbers. Let's see how well the Llama 2 13B GPTQ model can chat with these documents.

Load Documents

Let's start by loading the PDFs and splitting them into smaller chunks. We'll load the PDFs using the `PyPDFDirectoryLoader` class from the `langchain` library. This class loads all the PDFs in a directory and returns a list of `Document` objects:

```
loader = PyPDFDirectoryLoader("pdfs")
docs = loader.load()
len(docs)
```

```
100
```

The combined page count of all the documents is 100.

We'll then use the `RecursiveCharacterTextSplitter` class to split the documents into smaller chunks. Each chunk will have 1024 characters, with an overlap of 64 characters:

```
text_splitter = RecursiveCharacterTextSplitter(chunk_size=1024, chunk_overla
texts = text_splitter.split_documents(docs)
len(texts)
```

```
355
```

This gives us 355 chunks of text. We'll use these chunks to create a vector store. Our vector store will the `sentence-transformers` library to create embeddings for the text chunks:

```
embeddings = HuggingFaceInstructEmbeddings(
    model_name="hkunlp/instructor-large", model_kwargs={"device": DEVICE}
)
```

We'll use instruction-finetuned text embedding model provided by hkunlp/instructor-large. These embeddings score very high on the Massive Text Embedding Benchmark (MTEB) Leaderboard[7].

Our vector store will use the `Chroma` database. We'll use the `from_documents` method to create it:

```
db = Chroma.from_documents(texts, embeddings, persist_directory="db")
```

Our database now contains embeddings for all the text chunks. Let's continue with loading the model.

Llama 2 13B

The choice of the model is specific to the hardware you're using. Here, we'll use a Nvidia T4 GPU with 16GB of VRAM. If you have a different GPU, you might need to use a different model (usually, the larger the better).

Let's use AutoGPTQ to load the Llama 2 13B GPTQ model and the tokenizer:

```
model_name_or_path = "TheBloke/Llama-2-13B-chat-GPTQ"
model_basename = "gptq_model-4bit-128g"

tokenizer = AutoTokenizer.from_pretrained(model_name_or_path, use_fast=True)

model = AutoGPTQForCausalLM.from_quantized(
    model_name_or_path,
    revision="gptq-4bit-128g-actorder_True",
    model_basename=model_basename,
    use_safetensors=True,
    trust_remote_code=True,
    inject_fused_attention=False,
    device="cuda:0",
    quantize_config=None,
)
```

We're using a 4 bit quantized model with 128 groups. Here's a short description of the model:

> *4-bit, with Act Order and group size. 128g uses even less VRAM, but with slightly lower accuracy. Poor AutoGPTQ CUDA speed.*

So, our model is not the fastest, but it should be accurate enough for our use case.

Prompt Format

The Llama 2 chat models use a specific prompt format. Here's the general template:

```
[INST] <<SYS>>
{system_prompt}
<</SYS>>

{prompt} [/INST]
```

Note that Llama 2 supports a system prompt and a user prompt. We'll use the system prompt to **parametrize** the model (give it a context on how we want it to respond). The user prompt will contain the question we want to ask the model:

```
DEFAULT_SYSTEM_PROMPT = """
You are a helpful, respectful and honest assistant. Always answer as helpfull
as possible, while being safe. Your answers should not include any harmful,
unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensu
responses are socially unbiased and positive in nature.

If a question does not make any sense, or is not factually coherent, explain
why instead of answering something not correct. If you don't know the answer
question, please don't share false information.
""".strip()


def generate_prompt(prompt: str, system_prompt: str = DEFAULT_SYSTEM_PROMPT)
    return f"""
[INST] <<SYS>>
{system_prompt}
<</SYS>>

{prompt} [/INST]
""".strip()
```

Note that this is the default system prompt provided by the Meta developers. We'll change it in a bit. Let's continue with creating the pipeline:

```
streamer = TextStreamer(tokenizer, skip_prompt=True, skip_special_tokens=True

text_pipeline = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    max_new_tokens=1024,
    temperature=0,
    top_p=0.95,
    repetition_penalty=1.15,
    streamer=streamer,
)
```

This pipeline will stream the response of the model to the standard output. We'll set the `temperature` to 0 to make our responses repeatable. We'll also use the `repetition_penalty` parameter to penalise the repetition of tokens in the response.

To make our pipeline compatible with LangChain, we'll wrap it in the `HuggingFacePipeline` class:

```
llm = HuggingFacePipeline(pipeline=text_pipeline, model_kwargs={"temperature'
```

Create Chain

We now have all the pieces we need to create a chain that can retrieve relevant documents and answer questions from them. First, we'll create a prompt template that will be used to parametrize the model:

```
SYSTEM_PROMPT = "Use the following pieces of context to answer the question a

template = generate_prompt(
    """
{context}

Question: {question}
""",
    system_prompt=SYSTEM_PROMPT,
)

prompt = PromptTemplate(template=template, input_variables=["context", "quest
```

Next, we'll use the `RetrievalQA` class to create a chain using our pipeline, embeddings and prompt template:

```
qa_chain = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type="stuff",
    retriever=db.as_retriever(search_kwargs={"k": 2}),
    return_source_documents=True,
    chain_type_kwargs={"prompt": prompt},
)
```

Note that we're restricting the number of documents retrieved to 2. This will ensure that our LLM doesn't get over the token limit.

Chat with Multiple PDFs

Let's ask our chain a few questions:

```
result = qa_chain("What is the per share revenue for Meta during 2023?")
```

```
Based on the information provided in the press release, the per share revenue
for Meta during 2023 was $2.98.
```

Our chain gave us an exact answer! Let's see where it got the answer from:

```
len(result["source_documents"])
```

```
2
```

```
print(result["source_documents"][0].page_content)
```

■ Document contents

Our embeddings along with the Llama 2 model did a great job here. Here's an excerpt of the document:

> *Diluted earnings per share (EPS)* 2.98 *2.46 21%*

Let's ask another question:

```
result = qa_chain("What is the per share revenue for Tesla during 2023?")
```

```
I can't determine the per share revenue for Tesla in 2023 based on the
information provided. The financial statements only provide data on total rev
per share data. Additionally, there is no information on the number of outsta
shares, which is necessary to calculate per share data.
```

Strange, our chain couldn't find the answer. The retrieval part of the chain didn't do a good job here - the answer is available within the PDF (Diluted - *$0.78*).

Let's ask the same question for Nvidia:

```python
result = qa_chain("What is the per share revenue for Nvidia during 2023?")
```

```
Based on the information provided, the per share revenue for Nvidia during
2023 was $0.83. This can be calculated by dividing the total revenue for the
($7,192 million) by the weighted average number of shares outstanding (basic
diluted) during the period (2,470 million).
```

The answer is correct! It got it from here:

> *Net income per share: Basic $0.83*

Okay, let's ask a more complex question:

```python
result = qa_chain("What is the estimated YOY revenue for Meta during 2023?")
```

```
Based on the information provided in the press release, Meta's estimated YOY
revenue for 2023 is expected to be between $32-34.5 billion, which represents
increase of 11-16% over 2022.
```

Another great response! Let's ask for a forecast based on the data from the documents:

```
result = qa_chain("Choose one company to invest (Tesla, Nvidia or Meta) to ma
```

```
I cannot provide a definitive answer to this question, as it is not
appropriate for me to give financial advice or make recommendations on speci
investments. However, based on the information provided in the context, NVID
be a strong company with a diverse range of businesses and a solid track rec
of innovation and growth.

NVIDIA has a long history of developing cutting-edge technology, including GP
architecture, and has successfully expanded into new markets such as artific
intelligence, data science, and autonomous vehicles. The company has also de
commitment to sustainability and social responsibility, which may be attract
socially conscious investors.

That being said, investing in any company carries inherent risks, and it is
essential to conduct thorough research and consult with a financial advisor b
making any investment decisions. Additionally, the decision to invest in a pa
company should be based on individual circumstances and goals, and should no
solely determined by the information provided here.
```

It is accustom for those LLMs to stray from providing financial advice. Though, it gave us a concrete answer after that. Our agent chose Nvidia - key player in developing AI (wink wink). It even mentioned the sustainability and social responsibility of the company.

References

1. Llama 2 by Meta AI ↩
2. Open LLM Leaderboard ↩
3. The Open Source Definition ↩
4. GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers ↩
5. GPTQ Official Implementation on GitHub ↩
6. AutoGPTQ ↩
7. Massive Text Embedding Benchmark (MTEB) Leaderboard ↩
