MOF: Model Openness framework | XBRL: Data Collection for Fine-tuning

Daniel K., Jordyn Y., Henry T., Carina L., Oliver C., William F., Kaiwen H., Alex K., Marcus C., Christopher H., Vaibhav P., Hengrui R., Xin Chao C.

Model Openness Framework

LF AI & Data Blog

Introducing the Model Openness Framework: Promoting Completeness and Openness for Reproducibility, Transparency and Usability in AI

By cakerly | April 17, 2024 | No Comments

Prompt:
What licenses are recommended for Model Parameters under the Model Openness Framework?

Generation:
Under the Model Openness Framework, the recommended licenses for model parameters are the Apache 2.0 and MIT licenses.

Why a failure case:
The model response is misinformation. To date, model producers have been releasing model parameters (i.e., weights and biases) using an open source software license, such as Apache 2.0 and MIT, even though model parameters are not compatible with such licenses. As outlined in MOF, since model parameters are in fact data, model parameters should be distributed under an open data license, like CDLAPermissive-2.0.

Introduction

Overview:
This project investigates the Model Openness Framework (MOF) and XBRL to benchmark large language models (LLMs) in analyzing and extracting information from structured datasets.

Purpose:
-Explored how LLMs, like those from Hugging Face, handle real-world financial documents and datasets through Google Colab's accessible computing platform.

Key Areas of Study:
-Cleaning and expanding datasets from XBRL (e.g., SEC 10-K, 10-Q).
-Evaluating LLMs' performance using MOF benchmarks.
-Hands-on experimentation with financial and technical datasets.

Significance:
-By integrating MOF and XBRL, this project provides practical and technical insights into LLM capabilities, preparing students for applications in financial analytics and beyond.

Objectives

Skill Development:
-Hands-on experience in working with large language models (LLMs), including running and fine-tuning models using platforms like Google Colab.

Data Handling:
-Teach data cleaning and processing techniques for structured financial datasets, focusing on XBRL documents (e.g., SEC 10-K, 10-Q).

Model Benchmarking:
-Experiment with and benchmark LLMs using the Model Openness Framework (MOF) to evaluate their performance on financial data tasks.

Dataset Expansion:
-Build and expand datasets from XBRL documents to support future model training and testing.

Open Exploration:
-Test hypotheses, analyze results, and explore diverse use cases for LLMs in financial and technical domains.

Materials and Methods

Materials:

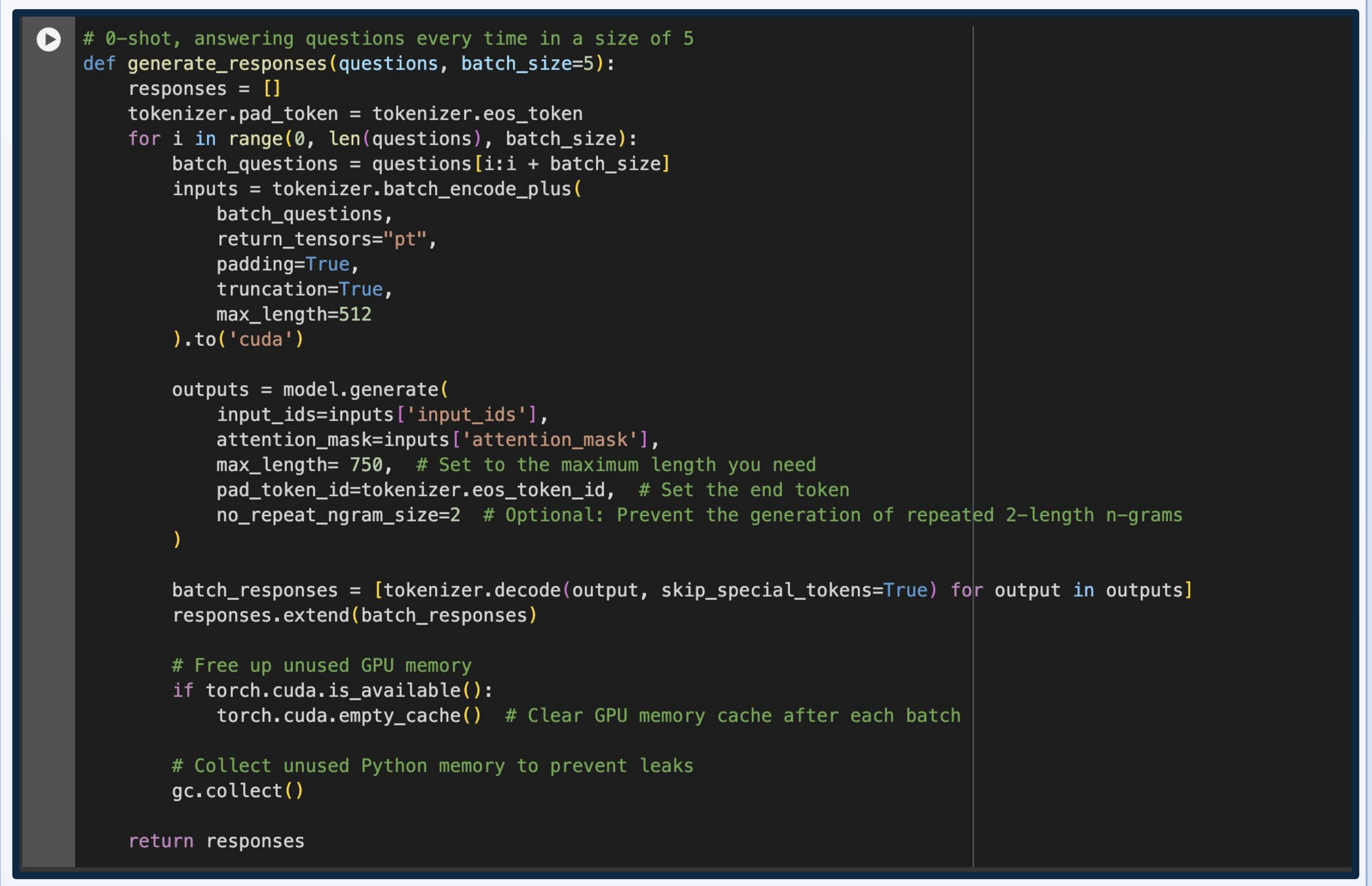
- **Google Colab**: Platform for running models with GPU access for large-scale computations.
- **LLMs (Hugging Face)**: Pre-trained models, including GPT-based and financial-specific models.
- **MOF Dataset**: Dataset used for evaluating LLMs' capabilities.
- **XBRL Data**: Structured financial documents (e.g., SEC 10-K, 10-Q) for dataset expansion and benchmarking.

Methods:

- **Model Execution**:
  Used Google Colab to run and test pre-trained LLMs for specific financial use cases.
- **Data Collection and Cleaning**:
  XBRL documents are collected, cleaned, and prepared for analysis using XML-based processing techniques.
- **Benchmarking LLMs**:
  LLMs are evaluated using the MOF dataset, with performance metrics compared across models.
- **Exploratory Analysis**:
  Open-ended experiments allow formulation of hypotheses and test LLM applications on customized datasets.
- **Collaboration and Documentation**:
  Code and results are documented in GitHub repositories to ensure reproducibility and foster collaboration

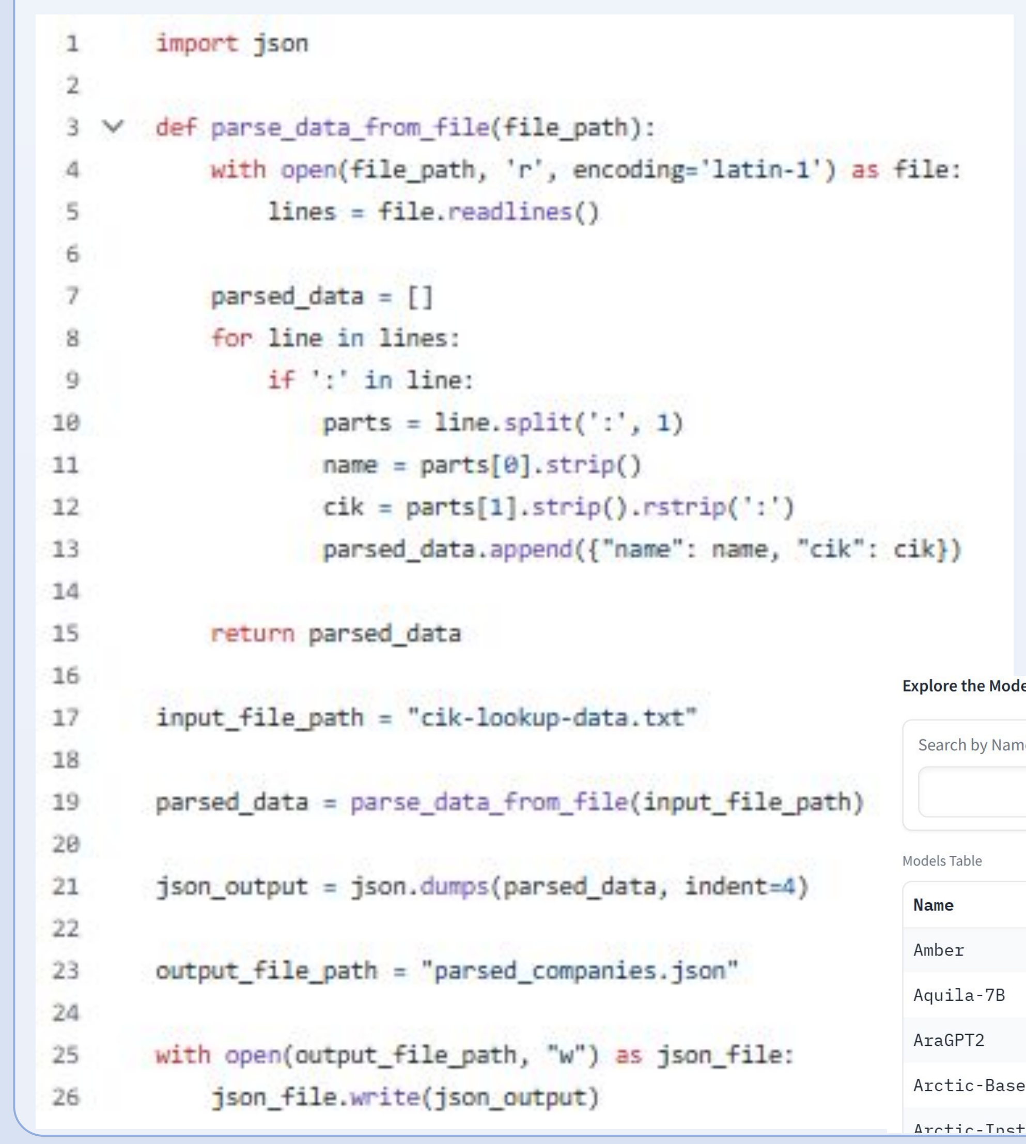
Category	Tasks	Progress	Results
CIK	Retrieve CIK numbers for all companies	Finished retrieving all CIK numbers and company names formatted into a json file.	.json file{ ciknumber, companyname, ...}
XBRL-filings	Download all XBRL filings -> 10K, 10Q, 8K	Code is complete with extensive testing on downloading each type of filing (5gb segment). The next step is to download in 500gb batches and upload it as a hugging face dataset.	Code complete. Next step: Download all filings
MOF	Hugging Face Space	In collaboration with the Linux Foundation, we helped set up the hugging face space for MOF (Model Openness Framework)	Still populating classification & badge.
Evaluation	Set up the evaluation pipeline for 200QA XBRL dataset	Finished writing the evaluation pipeline. The first step was extracting the corresponding xml test for the specific filing. The query and xml text were then inputted to the models for testing.	Continued testing of models.
Evaluation Debugging	Resolve the memory errors caused by the AI implementation.	Rewrote and modified and the majority of the evaluation code to accommodate for spacial/resource limitations using various HuggingFace models such as Gemma-3b and LLaMA-7b	The evaluation code began functioning under a smaller resource set.

XBRL Evaluation

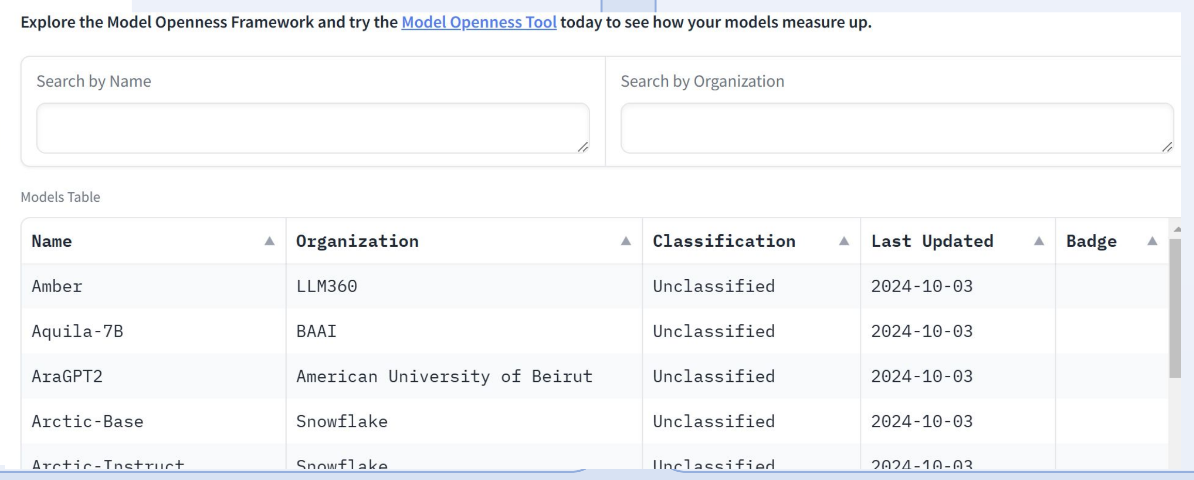

```python
# 0-shot, answering questions every time in a size of 5
def generate_responses(questions, batch_size=5):
    responses = []
    tokenizer.pad_token = tokenizer.eos_token
    for i in range(0, len(questions), batch_size):
        batch_questions = questions[i:i + batch_size]
        inputs = tokenizer.batch_encode_plus(
            batch_questions,
            return_tensors="pt",
            padding=True,
            truncation=True,
            max_length=512
        ).to('cuda')

        outputs = model.generate(
            input_ids=inputs['input_ids'],
            attention_mask=inputs['attention_mask'],
            max_length= 750,  # Set to the maximum length you need
            pad_token_id=tokenizer.eos_token_id,  # Set the end token
            no_repeat_ngram_size=2  # Optional: Prevent the generation of repeated 2-length n-grams
        )

        batch_responses = [tokenizer.decode(output, skip_special_tokens=True) for output in outputs]
        responses.extend(batch_responses)

        # Free up unused GPU memory
        if torch.cuda.is_available():
            torch.cuda.empty_cache()  # Clear GPU memory cache after each batch

        # Collect unused Python memory to prevent leaks
        gc.collect()

    return responses
```

- The XBRL Evaluation Pipeline is the labor-intensive task of financial data extraction, saving time and reducing errors. By delivering structured and contextual information, it helps users make informed decisions based on accurate and reliable financial data.
- The data gathered from web scraping the SEC database is converted into a JSON format. This JSON file will include items consisting of: a query, specific context for the query, and other additional instructions. This file will then be read item-by-item into an AI model which will use all previously mentioned information to return a correct response.

CIK


```python
import json

def parse_data_from_file(file_path):
    with open(file_path, 'r', encoding='latin-1') as file:
        lines = file.readlines()

    parsed_data = []
    for line in lines:
        if ':' in line:
            parts = line.split(':', 1)
            name = parts[0].strip()
            cik = parts[1].strip().rstrip(':')
            parsed_data.append({"name": name, "cik": cik})

    return parsed_data

input_file_path = "cik-lookup-data.txt"

parsed_data = parse_data_from_file(input_file_path)

json_output = json.dumps(parsed_data, indent=4)

output_file_path = "parsed_companies.json"

with open(output_file_path, "w") as json_file:
    json_file.write(json_output)
```

Parsed a large file of company names & CIK numbers separated by ":" using Python, then formatted all the final data into a json file.

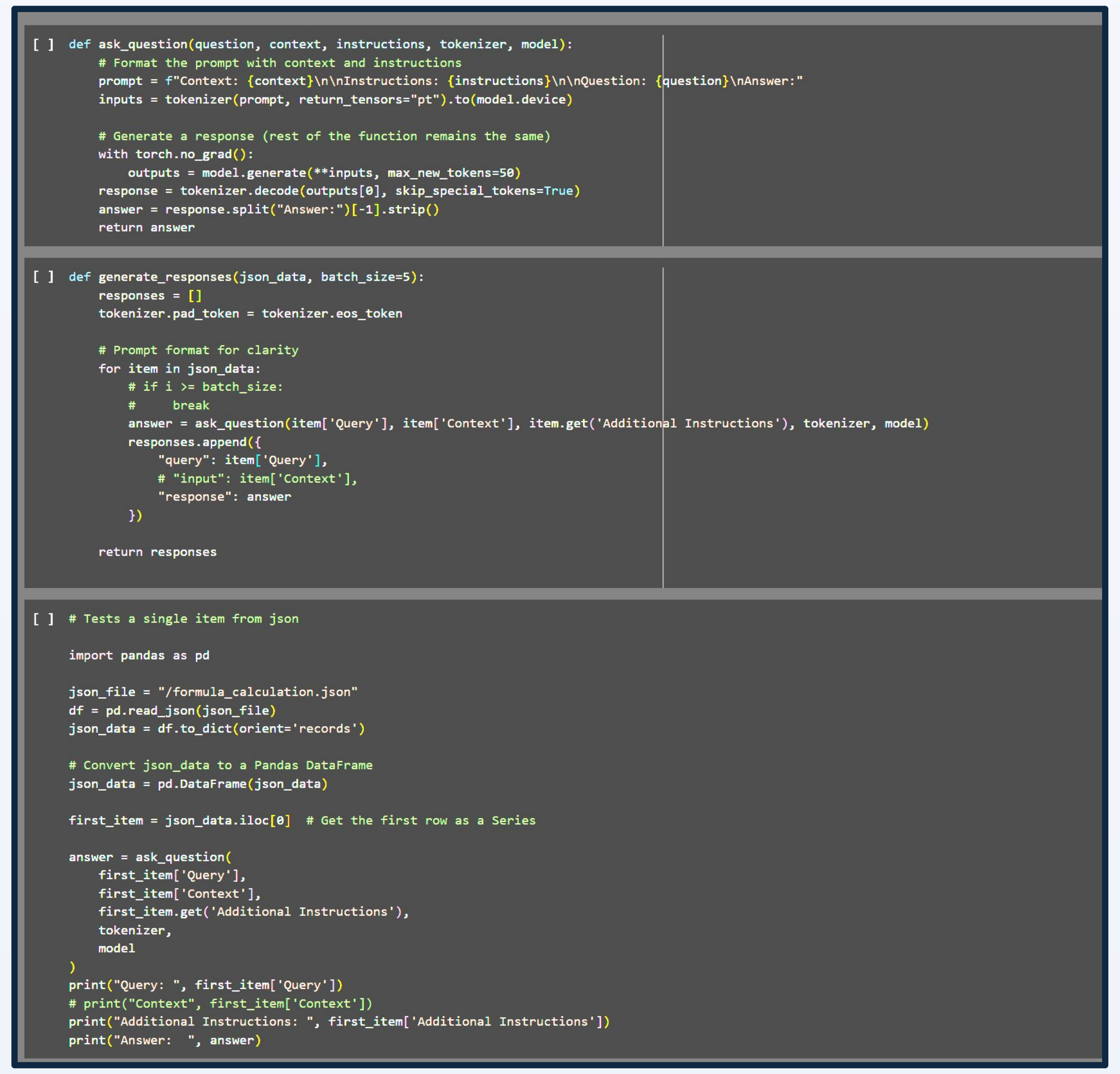

Explore the Model Openness Framework and try the Model Openness Tool today to see how your models measure up.

Search by Name | Search by Organization

Models Table

Name	Organization	Classification	Last Updated	Badge
Amber	LLM360	Unclassified	2024-10-03	
Aquila-7B	BAAI	Unclassified	2024-10-03	
AraGPT2	American University of Beirut	Unclassified	2024-10-03	
Arctic-Base	Snowflake	Unclassified	2024-10-03	
Arctic-Instruct	Snowflake	Unclassified	2024-10-03	

Processing


```python
[ ] def ask_question(question, context, instructions, tokenizer, model):
        # Format the prompt with context and instructions
        prompt = f"Context: {context}\n\nInstructions: {instructions}\n\nQuestion: {question}\nAnswer:"
        inputs = tokenizer(prompt, return_tensors="pt").to(model.device)

        # Generate a response (rest of the function remains the same)
        with torch.no_grad():
            outputs = model.generate(**inputs, max_new_tokens=50)
        response = tokenizer.decode(outputs[0], skip_special_tokens=True)
        answer = response.split("Answer:")[-1].strip()
        return answer
```

```python
[ ] def generate_responses(json_data, batch_size=5):
        responses = []
        tokenizer.pad_token = tokenizer.eos_token

        # Prompt format for clarity
        for item in json_data:
            # if i >= batch_size:
            #     break
            answer = ask_question(item['Query'], item['Context'], item.get('Additional Instructions'), tokenizer, model)
            responses.append({
                "query": item['Query'],
                # "input": item['Context'],
                "response": answer
            })

        return responses
```

```python
[ ] # Tests a single item from json

    import pandas as pd

    json_file = "/formula_calculation.json"
    df = pd.read_json(json_file)
    json_data = df.to_dict(orient='records')

    # Convert json_data to a Pandas DataFrame
    json_data = pd.DataFrame(json_data)

    first_item = json_data.iloc[0]  # Get the first row as a Series

    answer = ask_question(
        first_item['Query'],
        first_item['Context'],
        first_item.get('Additional Instructions'),
        tokenizer,
        model
    )
    print("Query: ", first_item['Query'])
    # print("Context", first_item['Context'])
    print("Additional Instructions: ", first_item['Additional Instructions'])
    print("Answer:  ", answer)
```

Conclusions

- Successfully retrieved **CIK numbers** and formatted company data into JSON, enabling streamlined dataset management.
- Completed and tested **XBRL filing downloads**, with plans to scale up to 5TB datasets for Hugging Face integration.
- Developed and debugged an **evaluation pipeline** for the 200QA XBRL dataset, optimizing for resource efficiency with models like Gemma-3b and LLaMA-7b.
- Contributed to the **Model Openness Framework (MOF)** by setting up Hugging Face space and advancing classification tasks.

References

- **GitHub - Chat XBRL**: Repository for exploring XBRL data and integrating LLMs. GitHub Chat XBRL
- **GitHub - Towards MOF**: Repository detailing the Model Openness Framework (MOF) setup and datasets. GitHub Towards MOF
- **Hugging Face Models**: Gemma-3b, LLaMA-7b, and other financial-specific LLMs for evaluation. Hugging Face
- **XBRL Filings**: SEC financial document types (10-K, 10-Q, 8-K) used for dataset generation. SEC.gov
- **Google Colab**: Platform for running and testing LLMs. Google Colab