# ASSIGNMENT 2: END-TO-END-NLP-SYSTEM BUILDING

## TO DO:

- You will *develop a retrieval augmented generation system (RAG)* (Lewis et al., 2021) that's capable of answering questions. You'll be working on the task of factual question-answering (QA). Since existing QA systems might not have the necessary knowledge in this domain, you will need to augment each question with relevant documents. Given an input question, your system will first retrieve documents and use those documents to generate an answer.
- You must collect your own data and develop a model of your choice on the data. You will run your already-constructed system over this data and submit the results. We also ask you to follow several experimental best practices, and describe the result in your report.
- Refer: thviet79/End-to-end-NLP-System-Building-ASM

## DETAIL:

The key checkpoints for this assignment are,

- Understand the task specification
- Prepare your raw data
- Annotate data for model development
- Develop a retrieval augmented generation system
- Generating results
- Write a report
- Submit your work

This is a group assignment, see the assignment policies for this class.[1]

## Task: Retrieval Augmented Generation (RAG)

You'll be working on the task of factual question-answering (QA). Since existing QA systems might not have the necessary knowledge in
this domain, you will need to augment each question with relevant documents. Given an input question, your system will first retrieve documents and use those documents to generate an answer.

Q: Who is Pittsburgh named after?
A: William Pitt

Q: What famous machine learning venue had its first conference in Pittsburgh in 1980?
A: ICML

Q: What musical artist is performing at PPG Arena on October 13?
A: Billie Eilish

## Data Format

Input (questions.txt): A text file containing one question per line.
Output (system_output.txt): A text file containing system generated answers. Each line contains a single answer string generated by your system for the corresponding question from questions.txt.
Reference (reference_answers.txt): A text file containing reference answers. Each line contains one or more reference answer strings for the corresponding question from questions.txt.
Read our model and data policy for this assignment.

## Preparing raw data

### Compiling a knowledge resource

For your test set and the RAG systems, you will first need to compile a knowledge resource of relevant documents. You are free to use any publicly available resource, but we *highly recommend* including the websites below. Note that we can also ask you questions from relevant subpages (e.g. "about", "schedule", "history", "upcoming events", "vendors", etc.) from these websites:

- General Introduction and History of VNU
  - Official Website of Vietnam National University, Hanoi (VNU): Provides an overview of the university's history, organizational structure, and mission.
  - Wikipedia page on VNU: Offers detailed information about the university's foundation and development over time.
- Admissions Information
  - VNU Admissions Portal: Includes information about academic programs, admission procedures, and entry requirements.
  - Admissions announcements from member universities: For example, USSH, ULIS, UET, HUS, ...
- Academic Regulations
  - Regulatory documents on education: Includes decisions and guidelines related to undergraduate and postgraduate training regulations.
  - Doctoral training regulations: Information about PhD-level training policies at VNU.
- Academic Programs
  - General training programs: List of undergraduate, master's, and doctoral programs offered at VNU.
  - International training programs: Information about joint training programs with international partners and programs taught in English.

## Collecting raw data

Your knowledge resource might include a mix of HTML pages, PDFs, and plain text documents. You will need to clean this data and convert it into a file format that suites your model development. Here are some tools that you could use:

- To process HTML pages, you can use beautifulsoup4.
- To parse PDF documents into plain text, you can use pypdf or pdfplumber.

By the end of this step, you will have a collection of documents that will serve as the knowledge resource for your RAG system.

### Annotating data

Next, you will want to annotate question-answer pairs for two purposes: testing/analysis and training. Use the documents you compiled in the previous step to identify candidate questions for annotation. You will then use the same set of documents to identify answers for your questions.

### Test data

The testing (and analysis) data will be the data that you use to make sure that your system is working properly. In order to do so, you will want to annotate enough data so that you can get an accurate estimate of how your system is doing, and if any improvements to your system are having a positive impact. Some guidelines on this,

- *Domain Relevance*: Your test data should be similar to the data that you will finally be tested on (questions about UET and VNU). Use the knowledge resources mentioned above to curate your test set.
- *Diversity*: Your test data should cover a wide range of questions UET and VNU.
- *Size*: Your test data should be large enough to distinguish between good and bad models. If you want some guidelines about this, see the lecture on experimental design and human annotation.[2]
- *Quality*: Your test data should be of high quality. We recommend that you annotate it yourself and validate your annotations within your team.

To help you get started, here are some example questions,

- Questions that could be answered by just prompting a LLM
  - When was Carnegie Mellon University founded?
- Questions that can be better answered by augmenting LLM with relevant documents
  - What is the name of the annual pickle festival held in Pittsburgh?
- Questions that are likely answered only through augmentation
  - When was the Pittsburgh Soul Food Festival established?
- Questions that are sensitive to temporal signals
  - Who is performing at X venue on Y date?

See Vu et al., 2023 for ideas about questions to prompt LLMs. For questions with multiple valid answers, you can include multiple reference answers per line in reference_answers.txt (separated by a semicolon;). As long as your system generates one of the valid answers, it will be considered correct.

This test set will constitute *data/test/questions.txt* and *data/test/reference_answers.txt* in your submission.

### Training data

The choice of training data is a bit more flexible, and depends on your implementation. If you are fine-tuning a model, you could possibly:

- Annotate it yourself manually through the same method as the test set.
- Do some sort of automatic annotation and/or data augmentation.
- Use existing datasets for transfer learning.

If you are using a LLM in a few-shot learning setting, you could possibly:

- Annotate examples for the task using the same method as the test set.
- Use existing datasets to identify examples for in-context learning.

This training set will constitute *data/train/questions.txt* and *data/train/reference_answers.txt* in your submission.

### Estimating your data quality

An important component of every data annotation effort is to estimate its quality. A standard approach is to measure inter-annotator agreement (IAA). To measure this, at least two members of your team should annotate a random subset of your test set. Compute IAA on this subset and report your findings.

## Developing your RAG system

Unlike assignment 1, there is no starter code for this assignment. You are *free to use any open-source model and library*, just make sure you provide due credit in your report. See our model policy.

For your RAG system, you will need the following three components,

1. Document & query embedder
2. Document retriever
3. Document reader (aka. question-answering system)

To get started, you can try langchain's RAG stack that utilizes GPT4All, Chroma and Llama2, as well as LlamaIndex.

Some additional resources that could be useful,

- 11711 lecture notes
- ACL 2023 tutorial on retrieval-augmented LMs
- llama-recipes for an example RAG chatbot with Llama2.
- Ollama or llama.cpp to run LLMs locally on your machine.

All the code for your data preprocessing, model development and evaluation will be a part of your GitHub repository (see submission for details).

### Generating results

Finally, you will run your systems on our test set (questions only) and submit your results to us. This test set will be released the day before the assignment is due.

### Unseen test set

This test set will be curated by the course staff and will evaluate your system's ability to respond to a variety of questions about Pittsburgh and CMU. Because the goal of this assignment is not to perform hyperparameter optimization on this private test set, we ask you to not overfit to this test set. You are allowed to submit up to *three* output files (system_outputs/system_output_{1,2,3}.txt). We will use the best performing file for grading.

## Evaluation metrics

Your submissions will be evaluated on standard metrics, answer recall, exact match and F1. See section 6.1 of the original SQuAD paper for details. These metrics are token-based and measure the overlap between your system answer and the reference answer(s). Therefore, we recommend keeping your system generated responses as concise as possible.

## Writing report

We ask you to write a report detailing various aspects about your end-to-end system development (see the grading criteria below).
There will be a 7 page limit for the report, and there is no required template. However, we encourage you to use the ACL template.
Make sure you cite all your sources (open-source models, libraries, papers, blogs etc.,) in your report.

## Submission & Grading

### Submission

Submit all deliverables on Canvas. Your submission checklist is below,

- Your report.
- A link to your GitHub repository containing your code.[3]
- A file listing contributions of each team member,
  - data annotation contributions from each team member (e.g. teammate A: instances 1-X; teammate B: instances X-Y, teammate C: instances Y-Z).
  - data collection (scraping, processing) and modeling contributions from each team member (e.g. teammate A: writing scripts to ..., implementing ...; teammate B:...; teammate C:...;)
- Testing and training data you annotated for this assignment.
- Your system outputs on our test set.

Your submission should be a zip file with the following structure (assuming the lowercase Andrew ID is ANDREWID). Make one submission per team.

```
ANDREWID/
├── report.pdf
├── github_url.txt
├── contributions.md
├── data/
│   ├── test/
│   │   ├── questions.txt
│   │   ├── reference_answers.txt
│   ├── train/
│   │   ├── questions.txt
│   │   ├── reference_answers.txt
├── system_outputs/
│   ├── system_output_1.txt
│   ├── system_output_2.txt (optional)
│   ├── system_output_3.txt (optional)
└── README.md
```

## Grading

The following points (max. 100 points) are derived from the results and your report. See course grading policy.[4]

- Submit data (15 points): submit testing/training data of your creation.
- Submit code (15 points): submit your code for preprocessing and model development in the form of a GitHub repo. We may not necessarily run your code, but we will look at it. So please ensure that it contains up-to-date code with a README file outlining the steps to run it. Your repo
- Results (30 points): points based on your system's performance on our private test set. 10 points for non-trivial performance,[5] plus up to 20 points based on level of performance relative to other submissions from the class.
- Report: below points are awarded based on your report.
  - Data creation (10 points): clearly describe how you created your data. Please include the following details,
    - How did you compile your knowledge resource, and how did you decide which documents to include?
    - How did you extract raw data? What tools did you use?
    - What data was annotated for testing and training (what kind and how much)?
    - How did you decide what kind and how much data to annotate?
    - What sort of annotation interface did you use?
    - How did you estimate the quality of your annotations? (IAA)
    - For training data that you did not annotate, did you use any extra data and in what way?

- Model details (10 points): clearly describe your model(s). Please include the following details,
  - What kind of methods (including baselines) did you try? Explain at least two variations (more is welcome). This can include which model you used, which data it was trained on, training strategy, etc.
  - What was your justification for trying these methods?
- Results (10 points): report raw numbers from your experiments. Please include the following details,
  - What was the result of each model that you tried on the testing data that you created?
  - Are the results statistically significant?
- Analysis (10 points): perform quantitative/qualitative analysis and present your findings,
  - Perform a comparison of the outputs on a more fine-grained level than just holistic accuracy numbers, and report the results. For instance, how did your models perform across various types of questions?
  - Perform an analysis that evaluates the effectiveness of retrieve-and-augment strategy vs closed-book use of your models.
  - Show examples of outputs from at least two of the systems you created. Ideally, these examples could be representative of the quantitative differences that you found above.

## Model and Data Policy

To make the assignment accessible to everyone,

- You are only allowed to use models that are also accessible through HuggingFace. This means you may *not* use closed models like OpenAI models, but you *can* opt to use a hosting service for an open model (such as the Hugging Face or Together APIs).
- You are only allowed to include publicly available data in your knowledge resource, test data and training data.
- You are welcome to use any open-source library to assist your data annotation and model development. Make sure you check the license and provide due credit.

## PHỤ LỤC

**Lưu ý một số bước khi chạy chương trình:**

Ví dụ về tập dữ liệu huấn luyện ở folder data/train/

1. File questions.txt

| |
|---|
| When was Carnegie Mellon University founded? |
| What is the name of the annual pickle festival held in Pittsburgh? |

   Mỗi câu nằm ở 1 dòng khác nhau

2. File reference_answers.txt

| |
|---|
| 1900 |
| Picklesburgh |

Lưu ý: Nếu có nhiều câu trả lời đúng cho một câu hỏi, hãy phân tách chúng bằng dấu chấm phẩy (;).

Ví dụ về tập dữ liệu test ở folder data/test/

1. File questions.txt

| |
|---|
| Who is Pittsburgh named after? |
| What famous machine learning venue had its first conference in Pittsburgh in 1980? |

   Mỗi câu nằm ở 1 dòng khác nhau

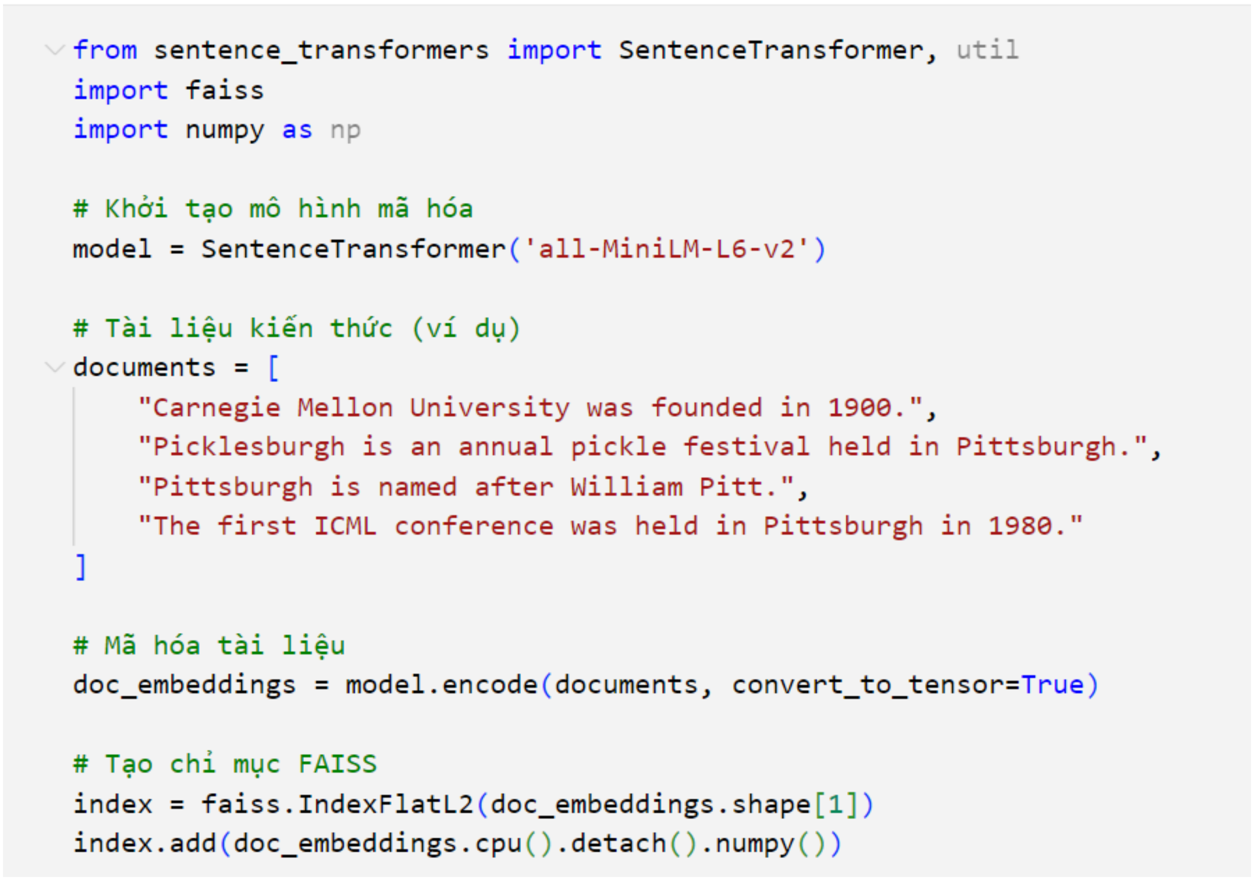
2. File reference_answers.txt

| |
|---|
| William Pitt |
| ICML |

Lưu ý: Nếu có nhiều câu trả lời đúng cho một câu hỏi, hãy phân tách chúng bằng dấu chấm phẩy (;).

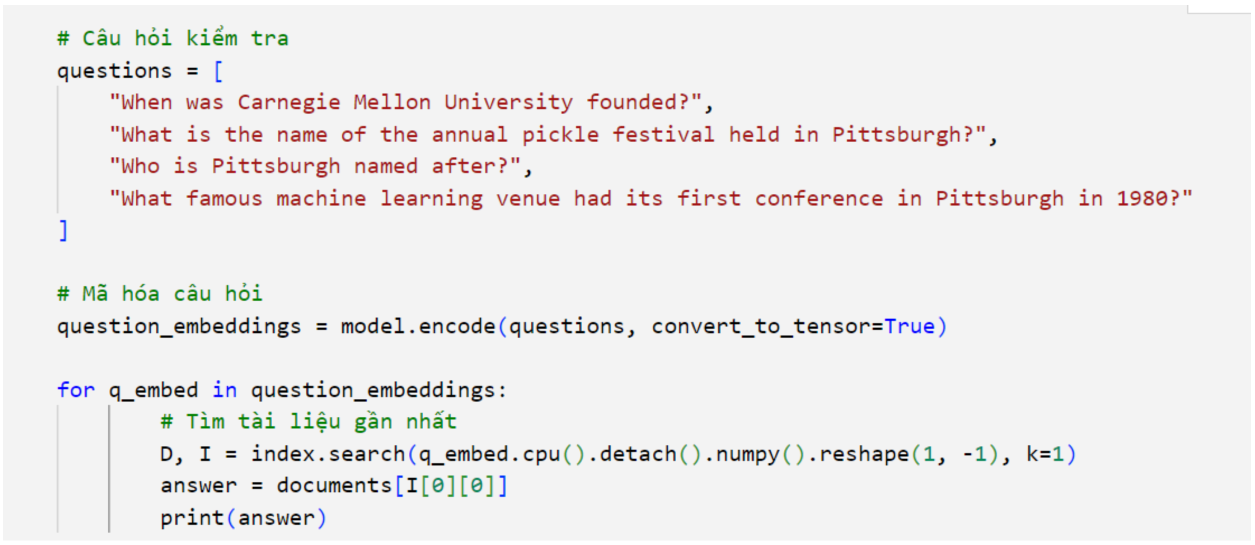
Sau khi có dữ liệu và ta biết được các câu hỏi này tham chiếu đến tài liệu nào

Giả sử ta có đoạn code ví dụ về các document và mô hình trích xuất (sinh viên có thể dùng mô hình và cách đánh chỉ mục khác không nhất thiết phải giống như ví dụ)

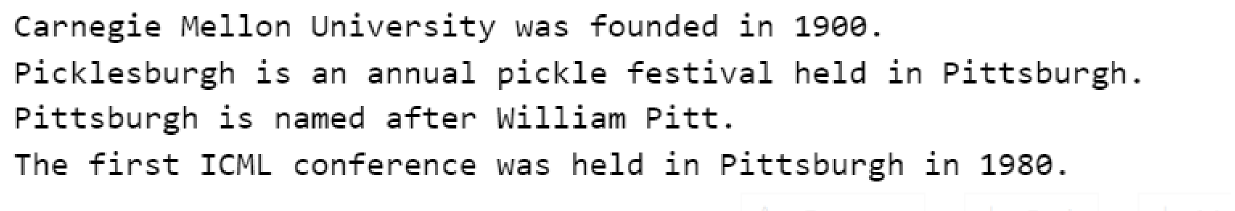


```python
from sentence_transformers import SentenceTransformer, util
import faiss
import numpy as np

# Khởi tạo mô hình mã hóa
model = SentenceTransformer('all-MiniLM-L6-v2')

# Tài liệu kiến thức (ví dụ)
documents = [
    "Carnegie Mellon University was founded in 1900.",
    "Picklesburgh is an annual pickle festival held in Pittsburgh.",
    "Pittsburgh is named after William Pitt.",
    "The first ICML conference was held in Pittsburgh in 1980."
]

# Mã hóa tài liệu
doc_embeddings = model.encode(documents, convert_to_tensor=True)

# Tạo chỉ mục FAISS
index = faiss.IndexFlatL2(doc_embeddings.shape[1])
index.add(doc_embeddings.cpu().detach().numpy())
```

Sau đó chúng ta có các câu hỏi

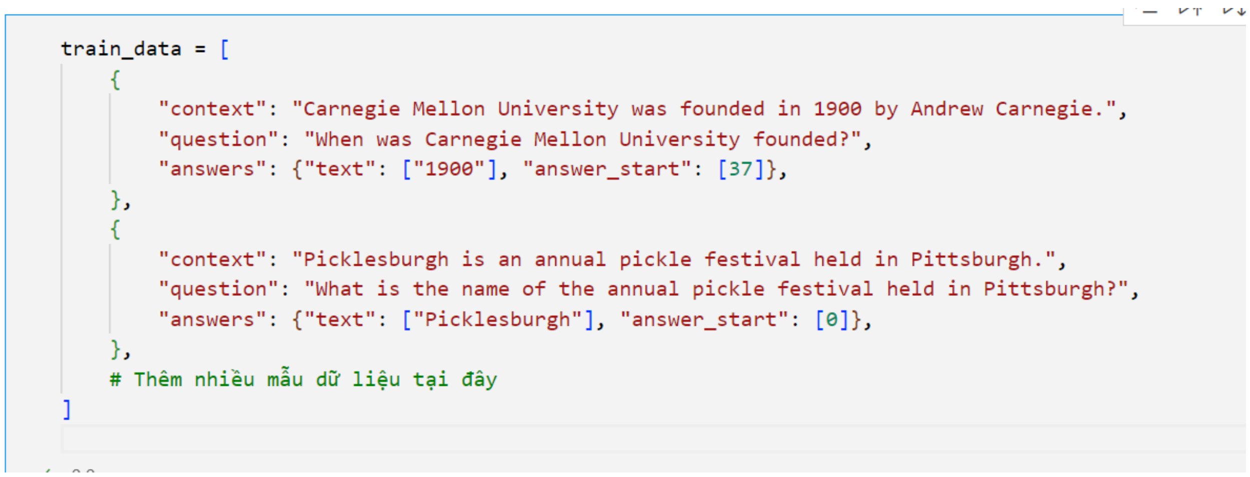


```python
# Câu hỏi kiểm tra
questions = [
    "When was Carnegie Mellon University founded?",
    "What is the name of the annual pickle festival held in Pittsburgh?",
    "Who is Pittsburgh named after?",
    "What famous machine learning venue had its first conference in Pittsburgh in 1980?"
]

# Mã hóa câu hỏi
question_embeddings = model.encode(questions, convert_to_tensor=True)

for q_embed in question_embeddings:
        # Tìm tài liệu gần nhất
        D, I = index.search(q_embed.cpu().detach().numpy().reshape(1, -1), k=1)
        answer = documents[I[0][0]]
        print(answer)
```



```
Carnegie Mellon University was founded in 1900.
Picklesburgh is an annual pickle festival held in Pittsburgh.
Pittsburgh is named after William Pitt.
The first ICML conference was held in Pittsburgh in 1980.
```

Và đưa ra được các document nào liên quan đến câu hỏi. Sau khi đưa được template liên quan đến câu hỏi có thể sử dụng một mô hình đưa vào câu hỏi và document liên quan vào để mô hình đưa ra câu trả lời.

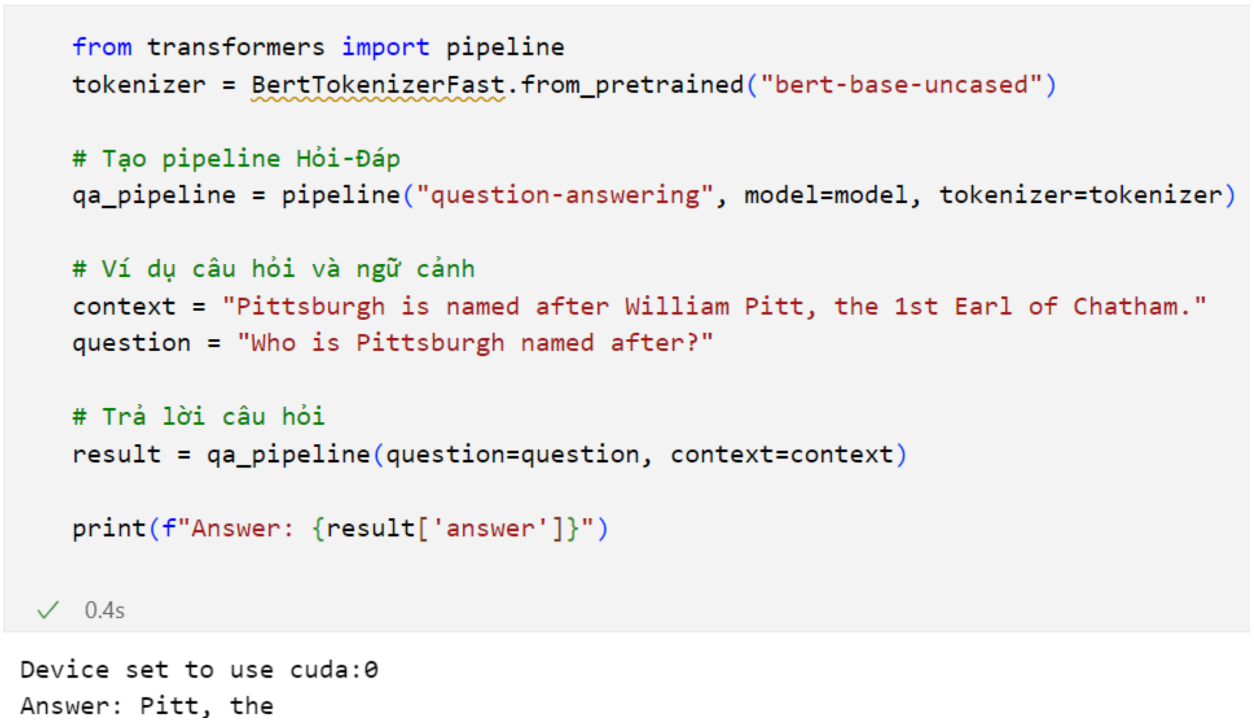


```python
train_data = [
    {
        "context": "Carnegie Mellon University was founded in 1900 by Andrew Carnegie.",
        "question": "When was Carnegie Mellon University founded?",
        "answers": {"text": ["1900"], "answer_start": [37]},
    },
    {
        "context": "Picklesburgh is an annual pickle festival held in Pittsburgh.",
        "question": "What is the name of the annual pickle festival held in Pittsburgh?",
        "answers": {"text": ["Picklesburgh"], "answer_start": [0]},
    },
    # Thêm nhiều mẫu dữ liệu tại đây
]
```

Ví dụ dữ liệu bên trên ta sẽ tạo được bộ dữ liệu cho việc training mô hình QA tham khảo ở link này

*https://huggingface.co/docs/transformers/en/tasks/question_ answering*

Sau khi train có thể thử infer (ví dụ mẫu là mô hình chưa train)



```python
from transformers import pipeline
tokenizer = BertTokenizerFast.from_pretrained("bert-base-uncased")

# Tạo pipeline Hỏi-Đáp
qa_pipeline = pipeline("question-answering", model=model, tokenizer=tokenizer)

# Ví dụ câu hỏi và ngữ cảnh
context = "Pittsburgh is named after William Pitt, the 1st Earl of Chatham."
question = "Who is Pittsburgh named after?"

# Trả lời câu hỏi
result = qa_pipeline(question=question, context=context)

print(f"Answer: {result['answer']}")
```

✓ 0.4s

```
Device set to use cuda:0
Answer: Pitt, the
```