

## ✓ COMS W4705 - Homework 4

### Question Answering with Retrieval Augmented Generation

Anubhav Jangra <[aj3228@columbia.edu](mailto:aj3228@columbia.edu)>, Emile Al-Billeh <[ea3048@columbia.edu](mailto:ea3048@columbia.edu)>, Daniel Bauer <[bauer@cs.columbia.edu](mailto:bauer@cs.columbia.edu)>

In this assignment, you will use a pretrained LLM for question answering on a subset of the Stanford QA Dataset (SQuAD). Here is an example question from SQuAD:

*Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?*

Specific domain knowledge to answer questions like this may not be available in the data that the LLM was pre-trained on. As a result, if we simply prompt the the LLM to answer this question, it may tell us that it does not know the answer, or worse, it may hallucinate an incorrect answer. Even if we are lucky and the LLM has have enough information to answer this question from pre-training, but the information may be outdated (the headmaster is likely to change from time to time).

Luckily, SQuAD provides a context snippet for each question that may contain the answer, such as

*The Christian Brothers of Ireland Stella Maris College is a private, co-educational, not-for-profit Catholic school located in the wealthy residential southeastern neighbourhood of Carrasco. Established in 1955, it is regarded as one of the best high schools in the country, blending a rigorous curriculum with strong extracurricular activities. **The school's headmaster, history professor Juan Pedro Toni, is a member of the Stella Maris Board of Governors and the school is a member of the International Baccalaureate Organization (IBO). Its long list of distinguished former pupils includes economists, engineers, architects, lawyers, politicians and even F1 champions. The school has also played an important part in the development of rugby union in Uruguay, with the creation of Old Christians Club, the school's alumni club.***

If we include the context as part of the prompt to the LLM, the model should be able to correctly answer the question (SQuAD contains "unanswerable questions", for which the provided context does not provide sufficient information to answer the question -- we will ignore these for the purpose of this assignment).

We will consider a scenario in which we don't know which context belongs to which question and we will use **Retrieval Augmented Generation (RAG)** techniques to identify the relevant context from the set of all available contexts.

Specifically we will experiment with the following systems:

- A baseline "vanilla QA" system in which we try to answer the question without any additional context (i.e. using the pre-trained LLM only).
- An "oracle" system, in which we provide the correct context for each question. This establishes an upper bound for the retrieval approaches.
- Two different approaches for retrieving relevant contexts:
  - based on token overlap between the question and each context.
  - based on cosine similarity between question embeddings and candidate context embeddings (obtained using BERT).

We will evaluate each system using a number of metrics commonly used for QA tasks:

- Exact Match (EM), which measures the percentage of predictions that exactly match the ground truth answers.
- F1 score, measured on the token overlap between the predicted and ground truth answers.
- ROUGE (specifically, ROUGE2)

Follow the instructions in this notebook step-by step. Much of the code is provided and just needs to be run, but some sections are marked with todo. Make sure to complete all these sections.

Requirements: Access to a GPU is required for this assignment. If you have a recent mac, you can try using mps. Otherwise, I recommend renting a GPU instance through a service like vast.ai or lambdalabs. Google Colab can work in a pinch, but you would have to deal with quotas and it's somewhat easy to lose unsaved work.

First, we need to ensure that transformers is installed, as well as the accelerate package.

```
!pip install transformers
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.12/dist-packages (4.57.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from transformers) (3.20.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (from transform
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2.0.2)
```

```

Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (6.0.3)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (2025)
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (from transform
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transformers) (0.7
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<1.0,
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.12/dist-packages (from hugginfac
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface-hub<
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->tr
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->transformers)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->transformer
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests->transfor

```

```
!pip install accelerate
```

```

Requirement already satisfied: accelerate in /usr/local/lib/python3.12/dist-packages (1.12.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from accelerate) (2.0.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from accelerate) (25.0)
Requirement already satisfied: psutil in /usr/local/lib/python3.12/dist-packages (from accelerate) (5.9.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.12/dist-packages (from accelerate) (6.0.3)
Requirement already satisfied: torch>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from accelerate) (2.9.0+cu126
Requirement already satisfied: huggingface_hub>=0.21.0 in /usr/local/lib/python3.12/dist-packages (from accelerate) (
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from accelerate) (0.7.0
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from huggingface_hub>=0.21.0->acc
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub>=0.2
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from huggingface_hub>=0.21.0->acc
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub>=0.21.0-
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.12/dist-packages (from hugginfac
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggingface_hub>
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate)
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerat
Requirement already satisfied: networkx>=2.5.1 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->acceler
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerate) (3.1
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from tor
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (from torch
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (from torch>=2
Requirement already satisfied: nvidia-cUBLAS-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (from torch>=2
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from torch>=2.
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (from torch>=
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (from torch>=
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (from torch>=
Requirement already satisfied: nvidia-cusparseLTL-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (from torch>=
Requirement already satisfied: nvidia-nccl-cu12==2.27.5 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0
Requirement already satisfied: nvidia-nvshmem-cu12==3.3.20 in /usr/local/lib/python3.12/dist-packages (from torch>=2.
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (from torch>=
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (from torch>=2
Requirement already satisfied: triton==3.5.0 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accelerat
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->tor
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->torch>=2.0.0-
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests->hu
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->huggingface_hu
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests->huggingf
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests->huggingf

```

Now all the relevant imports should succeed:

```

import os
import json
import tqdm
import copy
import torch
import torch.nn.functional as F

import re
import string
import collections

import transformers

```

## ▼ Data Preparation

This section creates the benchmark data we need to evaluate the QA systems. It has already been implemented for you. We recommend that you run it only once, save the benchmark data in a json file and then load it when needed. The following code may not work in Windows. We are providing the pre-generated benchmark data for download as an alternative.

```
data_dir = "./squad_data"
if not os.path.exists(data_dir):
    os.mkdir(data_dir)
```

## ▼ Downloading the Data and Creating the Benchmark Dataset

```
training_url = "https://rajpurkar.github.io/SQuAD-explorer/dataset/train-v2.0.json"
val_url = "https://rajpurkar.github.io/SQuAD-explorer/dataset/dev-v2.0.json"

os.system(f"curl -L {training_url} -o {data_dir}/squad_train.json")

0

# load the raw dataset
train_data = json.load(open(f"{data_dir}/squad_train.json"))

# Some details about the dataset

# SQuAD is split up into questions about a number of different topics
print(f"Number of topics: {len(train_data['data'])}")

# Let's explore just one topic. Each topic comes with a number of context paragraphs.
print("=*30)
print(f"For topic \"{{train_data['data'][0]['title']}}\"")
print(f"Number of available context paragraphs: {len(train_data['data'][0]['paragraphs'])}")
print("=*30)

print("The first paragraph is:")
print(train_data['data'][0]['paragraphs'][0]['context'])
print("=*30)

# Each paragraph comes with a number of question/answer pairs about the text in the paragraph
print("The first five question-answer pairs are:")
for qa in train_data['data'][0]['paragraphs'][0]['qas'][:10]:
    print(f"Question: {qa['question']}")
    print(f"Answer: {qa['answers'][0]['text']}")
    print("-*20)

Number of topics: 442
=====
For topic "Beyoncé"
Number of available context paragraphs: 66
=====
The first paragraph is:
Beyoncé Giselle Knowles-Carter (/bi:'jɒnseɪ/ bee-YON-say) (born September 4, 1981) is an American singer, songwriter,
=====
The first five question-answer pairs are:
Question: When did Beyonce start becoming popular?
Answer: in the late 1990s
-----
Question: What areas did Beyonce compete in when she was growing up?
Answer: singing and dancing
-----
Question: When did Beyonce leave Destiny's Child and become a solo singer?
Answer: 2003
-----
Question: In what city and state did Beyonce grow up?
Answer: Houston, Texas
-----
Question: In which decade did Beyonce become famous?
Answer: late 1990s
-----
Question: In what R&B group was she the lead singer?
Answer: Destiny's Child
-----
Question: What album made her a worldwide known artist?
Answer: Dangerously in Love
-----
Question: Who managed the Destiny's Child group?
Answer: Mathew Knowles
-----
Question: When did Beyoncé rise to fame?
```

```
Answer: late 1990s
```

```
-----  
Question: What role did Beyoncé have in Destiny's Child?
```

```
Answer: lead singer  
-----
```

```
print("Total number of paragraphs in the training set:", sum([len(topic['paragraphs']) for topic in train_data['data']])  
print("Total number of question-answer pairs in the training set:", sum([len(paragraph['qas']) for topic in train_da
```

```
Total number of paragraphs in the training set: 19035  
Total number of question-answer pairs in the training set: 130319
```

```
# not all questions are answerable given the information in the paragraph. Part of the original SQuAD 2 task is to i  
# unanswerable questions. We will ignore them for the purpose of this assignment.
```

```
print("Avg number of answers per question:",  
      sum([len(qa['answers']) for topic in train_data['data'] for paragraph in topic['paragraphs'] for qa in paragra  
           sum([len(paragraph['qas']) for topic in train_data['data'] for paragraph in topic['paragraphs']]))  
print("Count of answerable vs unanswerable questions:")
```

```
answerable_count = 0  
unanswerable_count = 0  
for topic in train_data['data']:  
    for paragraph in topic['paragraphs']:  
        for qa in paragraph['qas']:  
            if len(qa['answers']) > 0:  
                answerable_count += 1  
            else:  
                unanswerable_count += 1
```

```
print(f"Answerable questions: {answerable_count} ({answerable_count / (answerable_count + unanswerable_count) * 100:0.2f}%)"
```

```
print(f"Unanswerable questions: {unanswerable_count} ({unanswerable_count / (answerable_count + unanswerable_count) * 100:0.2f}%)"
```

```
Avg number of answers per question: 0.6662190471074824
```

```
Count of answerable vs unanswerable questions:
```

```
Answerable questions: 86821 (66.62%)
```

```
Unanswerable questions: 43498 (33.38%)
```

```
# Finally, create the RAG QA benchmark consisting of 250 answerable questions.
```

```
# We will use all available context paragraphs for RAG  
rag_contexts = [paragraph['context'] for topic in train_data['data'] for paragraph in topic['paragraphs']]
```

```
qa_pairs = []  
for topic in train_data['data']:  
    for paragraph in topic['paragraphs']:  
        for qa in paragraph['qas']:  
            if len(qa['answers']) > 0:  
                qa_pairs.append({  
                    "question": qa['question'],  
                    "answer": qa['answers'][0]['text'],  
                    "context": paragraph['context']  
                })
```

```
# randomly sample 250 answerable questions for the benchmark
```

```
import random
```

```
random.seed(42) # IMPORTANT so everyone is working on the same set of sampled QA pairs  
sampled_qa_pairs = random.sample(qa_pairs, 250)
```

```
evaluation_benchmark = {'qas': sampled_qa_pairs,  
                       'contexts': rag_contexts}  
random.shuffle(evaluation_benchmark['qas'])  
random.shuffle(evaluation_benchmark['contexts'])
```

```
# save the evaluation benchmark to a file  
json.dump(evaluation_benchmark, open(f"{data_dir}/rag_qa_benchmark.json", "w"), indent=2)
```

## >Loading the Benchmark Dataset / Understanding the Data Format

Use the following code to load the benchmark data from a file. Take a look at the example output to see how the data is structured.

```
# load the benchmark and display some samples  
evaluation_benchmark = json.load(open(f"{data_dir}/rag_qa_benchmark.json"))  
  
print("Sample RAG contexts:")  
for context in evaluation_benchmark['contexts'][:2]:
```

```

print(context)
print("-"*20)
print("="*30)
print("Sample RAG QA pairs:")
for qa in evaluation_benchmark['qas'][:5]:
    print(f"Question: {qa['question']}")"
    print(f"Answer: {qa['answer']}")"
    print("-"*20)

Sample RAG contexts:
Tajikistan's rivers, such as the Vakhsh and the Panj, have great hydropower potential, and the government has focused
-----
Two years later, the Emperor Valens, who favored the Arian position, in his turn exiled Athanasius. This time however
-----
=====

Sample RAG QA pairs:
Question: Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?
Answer: professor Juan Pedro Toni
-----
Question: What is the ratio of black and Asian schoolchildren to white schoolchildren?
Answer: about six to four
-----
Question: When did Outcault's The Yellow Kid appear in newspapers?
Answer: 1890s
-----
Question: When did devolution in the UK begin?
Answer: 1914
-----
Question: Treating the mitrailleuse like what rendered it far less effective
Answer: artillery
-----
```

The `evaluation_benchmark` is a dictionary with two keys:

- `evaluation_benchmark['qas']` provides a list of `qa_items` (see below).
- `evaluation_benchmark['contexts']` provides a list of available candidate contexts. Note that this includes all contexts from SQuAD, not just the ones for the 250 questions we sampled for the benchmark.

Each `qa_item` is a dictionary with the following keys:

- `qa_item['question']` is the question string
- `qa_item['answer']` is the target answer string
- `qa_item['context']` is the gold context for this question

For example:

```
qa_items = evaluation_benchmark['qas']
len(qa_items)
```

```
250
```

```
qa_item = qa_items[0]
qa_item['question']
```

```
'Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?'
```

```
qa_item['answer']
```

```
'professor Juan Pedro Toni'
```

```
qa_item['context']
```

```
'The Christian Brothers of Ireland Stella Maris College is a private, co-educational, not-for-profit Catholic school located in the wealthy residential southeastern neighbourhood of Carrasco. Established in 1955, it is regarded as one of the best high schools in the country, blending a rigorous curriculum with strong extracurricular activities. The school's headmaster, history professor Juan Pedro Toni, is a member of the Stella Maris Board of Governors and the school is a member of the International Baccalaureate Organization (IBO). Its long list of distinguished former pupils includes economists, engineers, architects, lawyers, politicians and even F1 champions. The school has also played'
```

## Part 1 - Question Answering Evaluation Functions

In this section, we will define a number of evaluation functions that measure the quality of the QA output, compared to a single target answer for each question.

Because the evaluation will happen at a token level, we will perform some simple pre-processing:

```
def normalize_answer(s):
    """Lower text and remove punctuation, articles and extra whitespace."""
    def remove_articles(text):
        regex = re.compile(r'\b(a|an|the)\b', re.UNICODE)
        return re.sub(regex, ' ', text)
    def white_space_fix(text):
        return ' '.join(text.split())
    def remove_punc(text):
        exclude = set(string.punctuation)
        return ''.join(ch for ch in text if ch not in exclude)
    def lower(text):
        return text.lower()
    return white_space_fix(remove_articles(remove_punc(lower(s)))))

def get_tokens(s):
    if not s: return []
    return normalize_answer(s).split()
```

First, Exact Match (EM) measures the percentage of predictions that match any one of the ground truth answers exactly after normalization. The following function returns 1 if the predicted answer is correct and 0 otherwise.

```
def compute_exact(a_gold, a_pred):
    return int(normalize_answer(a_gold) == normalize_answer(a_pred))
```

The next function calculates the  $F_1$  score of the set of predicted tokens against the set of target tokens.  $F_1$  is the harmonic mean of precision and recall, providing a balance between the two. Specifically

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

where precision is the fraction of predicted tokens that also appear in the target and recall is the fraction of target tokens that also appear in the prediction.

**TODO:** Write the function `compute_f1(a_gold, a_pred)` that returns the F1 score as defined above. It should work similar to the `compute_exact` method above. Test your function on a sample answer and prediction to verify that it works correctly.

```
def compute_f1(a_gold, a_pred): # Complete the function
    gold_tokens = get_tokens(a_gold)
    pred_tokens = get_tokens(a_pred)

    if len(gold_tokens) == 0 or len(pred_tokens) == 0:
        return 0.0

    common = set(gold_tokens) & set(pred_tokens)
    if len(common) == 0:
        return 0.0

    precision = len(common) / len(pred_tokens)
    recall = len(common) / len(gold_tokens)

    f1 = 2 * precision * recall / (precision + recall)
    return f1
```

```
# Test your function
reference_answers = ["London", "The capital of England is London.", "London is the capital city of England."]
predicted_answers = ["London, capital of England"] * len(reference_answers)

for ref, pred in zip(reference_answers, predicted_answers):
    print(f"Original:")
    print(f"Reference: {ref} | Predicted: {pred}")
    print(f"Normalized:")
    print(f"Reference: {normalize_answer(ref)} | Predicted: {normalize_answer(pred)}")
    print("Exact Match:", compute_exact(normalize_answer(ref), normalize_answer(pred)))
    print("F1 Score:", compute_f1(normalize_answer(ref), normalize_answer(pred)))
    print("-"*40)
```

```
Original:
Reference: London | Predicted: London, capital of England
Normalized:
Reference: london | Predicted: london capital of england
```

```

Exact Match: 0
F1 Score: 0.4
-----
Original:
Reference: The capital of England is London. | Predicted: London, capital of England
Normalized:
Reference: capital of england is london | Predicted: london capital of england
Exact Match: 0
F1 Score: 0.88888888888889
-----
Original:
Reference: London is the capital city of England. | Predicted: London, capital of England
Normalized:
Reference: london is capital city of england | Predicted: london capital of england
Exact Match: 0
F1 Score: 0.8
-----
```

Finally, we are also want to compute ROUGE-2 scores (which extends the F1 score above to 2-grams). We can use the `rouge_score` package to do this for us.

```

!pip install rouge_score

Collecting rouge_score
  Downloading rouge_score-0.1.2.tar.gz (17 kB)
    Preparing metadata (setup.py) ... done
Requirement already satisfied: absl-py in /usr/local/lib/python3.12/dist-packages (from rouge_score) (1.4.0)
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (from rouge_score) (3.9.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.12/dist-packages (from rouge_score) (2.0.2)
Requirement already satisfied: six>=1.14.0 in /usr/local/lib/python3.12/dist-packages (from rouge_score) (1.17.0)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk->rouge_score) (8.3.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk->rouge_score) (1.5.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk->rouge_score) (2
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk->rouge_score) (4.67.1)
Building wheels for collected packages: rouge_score
  Building wheel for rouge_score (setup.py) ... done
  Created wheel for rouge_score: filename=rouge_score-0.1.2-py3-none-any.whl size=24934 sha256=2f0c5d973b1b4f54088344
  Stored in directory: /root/.cache/pip/wheels/85/9d/af/01feefbe7d55ef5468796f0c68225b6788e85d9d0a281e7a70
Successfully built rouge_score
Installing collected packages: rouge_score
Successfully installed rouge_score-0.1.2
```

```

from rouge_score import rouge_scorer

rouge_scorer = rouge_scorer.RougeScorer(['rouge2'], use_stemmer=False)

def compute_rouge2(a_gold, a_pred):
    if not a_gold or not a_pred:
        return 0.0
    scores = rouge_scorer.score(a_gold.lower(), a_pred.lower())
    return scores['rouge2'].fmeasure
```

Let's test the metrics:

```

reference_answers = ["London", "The capital of England is London.", "London is the capital city of England."]
predicted_answers = ["London, capital of England"] * len(reference_answers)

print("Normalized Answers:")
for ref, pred in zip(reference_answers, predicted_answers):
    print(f"Original:")
    print(f"Reference: {ref} | Predicted: {pred}")
    print(f"Normalized:")
    print(f"Reference: {normalize_answer(ref)} | Predicted: {normalize_answer(pred)}")
    print("Exact Match:", compute_exact(normalize_answer(ref), normalize_answer(pred)))
    print("F1 Score:", compute_f1(normalize_answer(ref), normalize_answer(pred)))
    print("ROUGE-2 F1-score:", compute_rouge2(normalize_answer(ref), normalize_answer(pred)))
    print("-"*40)

Normalized Answers:
Original:
Reference: London | Predicted: London, capital of England
Normalized:
Reference: london | Predicted: london capital of england
Exact Match: 0
F1 Score: 0.4
ROUGE-2 F1-score: 0.0
```

```
-----  
Original:  
Reference: The capital of England is London. | Predicted: London, capital of England  
Normalized:  
Reference: capital of england is london | Predicted: london capital of england  
Exact Match: 0  
F1 Score: 0.88888888888889  
ROUGE-2 F1-score: 0.5714285714285715  
-----
```

```
Original:  
Reference: London is the capital city of England. | Predicted: London, capital of England  
Normalized:  
Reference: london is capital city of england | Predicted: london capital of england  
Exact Match: 0  
F1 Score: 0.8  
ROUGE-2 F1-score: 0.25  
-----
```

## ▼ Part 2 - Vanilla Question Answering

In this part, we will use an off-the-shelf pretrained LLM and attempt to answer the questions from its pretraining knowledge only. To make things simple, we will use the huggingface transformer pipeline abstraction. The pipeline will download the model and parameters for us on creation. When we pass an input prompt to the pipeline, it will automatically perform preprocessing (tokenization), inference, and postprocessing (removing EOS markers and padding).

### ▼ Loading the LLM

The LLM we will use is the 1B version of the instruction tuned OLMo2 model. OLMo is an open source language model created by Allen AI and the University of Washington. Unlike other open source models, OLMo is also open data. You can read more about it here: <https://huggingface.co/allenai/OLMo-2-0425-1B-Instruct> and here <https://allenai.org/olmo>.

```
qa_model = "allenai/OLMo-2-0425-1B-Instruct"  
  
from transformers import pipeline  
  
# Check which GPU device to use. Note, this will likely NOT work on a CPU.  
if torch.cuda.is_available():  
    device = "cuda"  
elif torch.backends.mps.is_available():  
    device = "mps"  
else:  
    device = "cpu"  
  
pipe = pipeline(  
    "text-generation",  
    model=qa_model,  
    dtype=torch.bfloat16,  
    device_map=device,  
)  
  
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: 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/token).  
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%                                         625/625 [00:00<00:00, 78.9kB/s]  
model.safetensors: 100%                                     2.97G/2.97G [00:08<00:00, 111MB/s]  
generation_config.json: 100%                                121/121 [00:00<00:00, 15.3kB/s]  
tokenizer_config.json: 4.88k/? [00:00<00:00, 557kB/s]  
vocab.json: 1.61M/? [00:00<00:00, 69.0MB/s]  
merges.txt: 917k/? [00:00<00:00, 54.6MB/s]  
tokenizer.json: 7.14M/? [00:00<00:00, 168MB/s]  
special_tokens_map.json: 100%                               581/581 [00:00<00:00, 78.6kB/s]  
Device set to use cuda
```

We can now pass a prompt to the model and retrieve the completed answer.

```
prompt = "My favorite thing to do in fall is"
output = pipe(prompt,
             max_new_tokens=128,
             do_sample=True, # set to False for greedy decoding below
             pad_token_id=pipe.tokenizer.eos_token_id)
print(output)

[{'generated_text': "My favorite thing to do in fall is bake homemade bread. Would you like to come over for dinner t"}]
```

We can skip the prompt that is repeated in the output.

```
output[0]['generated_text'][len(prompt):].strip()

'bake homemade bread. Would you like to come over for dinner tonight?\n\nWould you like to know what we're having? I've been experimenting with some new recipes and I'd be thrilled to show you some of my favorites.\n\nPlease tell me about your favorite fall dishes.\n\nWould you like to try some of my homemade cranberry sauce with the roasted turkey? I've been making it every year and it's always a hit.\n\nFeel free to ask me anything else about fall baking or
```

## Using the LLM for Question Answering

**TODO:** Write a function `vanilla_qa(qa_item)` that takes a `qa_item` in the format described above, inserts the question (and only the question!) into a suitable prompt, passes the prompt to the LLM and then returns the answer as a string.

A prompt might look like this, but will need a bit of prompt engineering to make it work well.

*Answer the following question concisely.*

*Question: Who played the lead role in Alien?*

*Answer:*

Once you have a basic version of the vanilla QA system you can tune the prompt (see below).

```
def vanilla_qa(qa_item): # Complete this function
    prompt = (
        "As a factual question answering system, you are going to answer a question.\n"
        "Answer the question using as few words as possible.\n"
        "Do not include articles (a, an, the), punctuation, or explanations.\n\n"
        "Example:\n"
        "Question: What year is this year?\n"
        "Answer: 2025\n"
        f"Question: {qa_item['question']}?\n"
        "Answer:\n"
    )
    output = pipe(prompt,
                  max_new_tokens=128,
                  do_sample=True, # set to False for greedy decoding below
                  pad_token_id=pipe.tokenizer.eos_token_id)
    answer = output[0]['generated_text'][len(prompt):].strip()
    return answer
```

The following code should return an answer (but possibly not the right one) to the first question in the dataset.

```
qa_item = evaluation_benchmark['qas'][0]
qa_item['question']
qa_item['answer']

'professor Juan Pedro Toni'
```

```
vanilla_qa(qa_item) # inspect the item

'Ian K Folly'
```

And the following function evaluates the performance of your `vanilla_qa` function on a list of `qa_items`.

```

def evaluate_qa(qa_function, qa_items, verbose=False):
    results = []

    for i, qa_item in tqdm.tqdm(enumerate(qa_items), desc="Evaluating QA instances", total=len(qa_items)):

        question = qa_item['question']
        answer = qa_item['answer']
        context = qa_item['context']

        predicted_answer = qa_function(qa_item)

        exact_match = compute_exact(answer, predicted_answer)
        f1_score = compute_f1(answer, predicted_answer)
        rouge2_f1 = compute_rouge2(answer, predicted_answer)

        if verbose:
            print(f"Q: {question}")
            print(f"Gold Answer: {answer}")
            print(f"Predicted Answer: {predicted_answer}")
            print(f"Exact Match: {exact_match}, F1 Score: {f1_score}")
            print(f"ROUGE-2 F1 Score: {rouge2_f1}")
            print("-"*40)

        results.append({
            "question": question,
            "answer": answer,
            "predicted_answer": predicted_answer,
            "context": context if context else None,
            "exact_match": exact_match,
            "f1_score": f1_score,
            "rouge2_f1": rouge2_f1
        })
    return results

```

```
vanilla_evaluation_results = evaluate_qa(vanilla_qa, evaluation_benchmark['qas'])
```

```
Evaluating QA instances: 100%|██████████| 250/250 [00:39<00:00, 6.30it/s]
```

```
vanilla_evaluation_results[0]
```

```
{
    'question': 'Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?',
    'answer': 'professor Juan Pedro Toni',
    'predicted_answer': 'Brendan Comer',
    'context': "The Christian Brothers of Ireland Stella Maris College is a private, co-educational, not-for-profit Catholic school located in the wealthy residential southeastern neighbourhood of Carrasco. Established in 1955, it is regarded as one of the best high schools in the country, blending a rigorous curriculum with strong extracurricular activities. The school's headmaster, history professor Juan Pedro Toni, is a member of the Stella Maris Board of Governors and the school is a member of the International Baccalaureate Organization (IBO). Its long list of distinguished former pupils includes economists, engineers, architects, lawyers, politicians and even F1 champions. The school has also played an important part in the development of rugby union in Uruguay, with the creation of Old Christians Club, the school's alumni club.",
    'exact_match': 0,
    'f1_score': 0.0,
    'rouge2_f1': 0.0
}
```

The function returns a list of evaluation results, one dictionary for each qa item.

Finally, the `present_results` function aggregates the results for the various qa items and prints the overall result.

```

def present_results(eval_results, exp_name=""):
    print(f"{exp_name} Evaluation Results:")
    exact_matches = [res['exact_match'] for res in eval_results]
    f1_scores = [res['f1_score'] for res in eval_results]
    rouge2_f1 = [res['rouge2_f1'] for res in eval_results]
    print(f"Exact Match: {sum(exact_matches) / len(exact_matches) * 100:.2f}%")
    print(f"F1 Score: {sum(f1_scores) / len(f1_scores) * 100:.2f}%")
    print(f"ROUGE2 F1: {sum(rouge2_f1) / len(rouge2_f1) * 100:.2f}%")

    # print out some evaluation results
    for res in eval_results[:5]:
        print(f"Question: {res['question']}")
        print(f"Gold Answer: {res['answer']}")
        print(f"Predicted Answer: {res['predicted_answer']}")
        print(f"Exact Match: {res['exact_match']}, F1 Score: {res['f1_score']}")

```

```

print("ROUGE-2 F1-score:", res['rouge2_f1'])
print("-"*40)

```

```
present_results(vanilla_evaluation_results, "Vanilla QA")
```

Vanilla QA Evaluation Results:

Exact Match: 6.40%

F1 Score: 12.12%

ROUGE2 F1: 2.26%

Question: Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?

Gold Answer: professor Juan Pedro Toni

Predicted Answer: Brendan Comer

Exact Match: 0, F1 Score: 0.0

ROUGE-2 F1-score: 0.0

---

Question: What is the ratio of black and Asian schoolchildren to white schoolchildren?

Gold Answer: about six to four

Predicted Answer: 3:2

Exact Match: 0, F1 Score: 0.0

ROUGE-2 F1-score: 0.0

---

Question: When did Outcault's The Yellow Kid appear in newspapers?

Gold Answer: 1890s

Predicted Answer: 1895

Exact Match: 0, F1 Score: 0.0

ROUGE-2 F1-score: 0.0

---

Question: When did devolution in the UK begin?

Gold Answer: 1914

Predicted Answer: Scotland

Exact Match: 0, F1 Score: 0.0

ROUGE-2 F1-score: 0.0

---

Question: Treating the mitrailleuse like what rendered it far less effective

Gold Answer: artillery

Predicted Answer: French

Exact Match: 0, F1 Score: 0.0

ROUGE-2 F1-score: 0.0

**TODO:** Experiment with the prompt template and try to achieve an Exact Match score of at least 5%. You may want to try including an example in the prompt (single-shot prompting).

## ▼ Part 3 - Oracle Question Answering

We will now establish an upper bound for a retrieval augmented QA system by providing the correct ("gold") context for each question as part of the prompt. These contexts are available as part of each qa\_item in the evaluation\_benchmark['qas'] dictionary.

**TODO:** Write a function `oracle_qa(qa_item)` that takes in a qa\_item, inserts both the question **and** the gold context into a prompt template, then passes the prompt to the LLM and returns the answer. The function should behave like the `vanilla_qa` function above, so that we can evaluate it using the same evaluation steps.

```

def oracle_qa(qa_item): # Write this function
    prompt = (
        f"As a factual question answering system, you are going to answer question.\n"
        f"Answer the question using as few words as possible.\n"
        f"Do not include articles (a, an, the), punctuation, or explanations.\n"
        f"Context: {qa_item['context']}"
        f"Question: {qa_item['question']}\\n"
        f"Answer:\\n"
    )

    output = pipe(prompt,
                  max_new_tokens=128,
                  do_sample=True, # set to False for greedy decoding below
                  pad_token_id=pipe.tokenizer.eos_token_id)

    answer = output[0]['generated_text'][len(prompt):].strip()
    return answer

```

**TODO:** run the `evaluate_qa` function on your `oracle_qa` function and display the results. You should see Exact Match scores above 50% (if not, tinker with the prompt template).

```

oracle_evaluation_results = evaluate_qa(oracle_qa, evaluation_benchmark['qas'])
present_results(oracle_evaluation_results)

Evaluating QA instances: 100%|██████████| 250/250 [00:32<00:00,  7.70it/s] Evaluation Results:
Exact Match: 56.80%
F1 Score: 68.60%
ROUGE2 F1: 32.01%
Question: Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?
Gold Answer: professor Juan Pedro Toni
Predicted Answer: Juan Pedro Toni
Exact Match: 0, F1 Score: 0.8571428571428571
ROUGE-2 F1-score: 0.8
-----
Question: What is the ratio of black and Asian schoolchildren to white schoolchildren?
Gold Answer: about six to four
Predicted Answer: 6 4
Exact Match: 0, F1 Score: 0.0
ROUGE-2 F1-score: 0.0
-----
Question: When did Outcault's The Yellow Kid appear in newspapers?
Gold Answer: 1890s
Predicted Answer: 1890s
Exact Match: 1, F1 Score: 1.0
ROUGE-2 F1-score: 0.0
-----
Question: When did devolution in the UK begin?
Gold Answer: 1914
Predicted Answer: 1914
Exact Match: 1, F1 Score: 1.0
ROUGE-2 F1-score: 0.0
-----
Question: Treating the mitrailleuse like what rendered it far less effective
Gold Answer: artillery
Predicted Answer: artillery
Exact Match: 1, F1 Score: 1.0
ROUGE-2 F1-score: 0.0
-----
```

## ▼ Part 4 - Retrieval-Augmented Question Answering - Word Overlap

Next, we will experiment with various approaches for retrieving relevant contexts from the set of available contexts. We first get the list of all 19035 available candidate contexts from the evaluation\_benchmark.

```
candidate_contexts = evaluation_benchmark["contexts"]
```

```
len(candidate_contexts)
```

```
19035
```

```
candidate_contexts[0]
```

```
'Tajikistan's rivers, such as the Vakhsh and the Panj, have great hydropower potential, and the government has focused on attracting investment for projects for internal use and electricity exports. Tajikistan is home to the Nurek Dam, the highest dam in the world. Lately, Russia's RAO UES energy giant has been working on the Sangtuda-1 hydroelectric power station (670 MW capacity) commenced operations on 18 January 2008. Other projects at the development stage include Sangtuda-2 by Iran, Zerafshan by the Chinese company SinoHydro, and the Rogun power plant that, at a projected height of 335 metres (1,099 ft), would supersede the Nurek Dam as highest in the world if it is brought to completion. A planned project, CASA 1000, will transmit 1000 MW of surplus electricity from Tajikistan to Pakistan with'
```

## ▼ Token Overlap Retriever

Let's first experiment with a simple retriever based on word overlap. Given a question, we measure how many of its tokens appear in each of the candidate contexts. We then retrieve the k contexts with the highest overlap.

**TODO:** Write the function `retrieve_overlap(question, contexts, top_k=5)` that takes in the question (a string) and a list of contexts (each context is a string). It should calculate the word overlap between the question and *each* context, and return a list of the *top\_k* contexts with the highest overlap.

```
# word overlap retriever -- write this function
def retrieve_overlap(question, contexts, top_k=5):
    question_tokens = set(get_tokens(question))
```

```

scores = []
for context in contexts:
    context_tokens = set(get_tokens(context))
    overlap = len (set(question_tokens) & set(context_tokens))
    scores.append((overlap, context))

scores.sort(key=lambda x: x[0], reverse=True)

return [context for _, context in scores[:top_k]]

```

The following function runs the retriever a list of qa\_items. For each qa\_item it obtains the list of retrieved contexts and adds them to the qa\_item.

```

def add_rag_context_overlap(qa_items, contexts, retriever, top_k=5):
    result_items = copy.deepcopy(qa_items)
    for inst in tqdm.tqdm(result_items, desc="Retrieving contexts"):
        question = inst['question']
        retrieved_contexts = retriever(question, contexts, top_k)
        inst['rag_contexts'] = retrieved_contexts
    return result_items

```

```

rag_qa_pairs = add_rag_context_overlap(evaluation_benchmark['qas'], candidate_contexts, retrieve_overlap)

Retrieving contexts: 100%|██████████| 250/250 [08:14<00:00,  1.98s/it]

```

It returns a copy of the qa\_item list that is now annotated with the additional 'rag\_contexts'.

```

rag_qa_pairs[0]

{'question': 'Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?', 'answer': 'professor Juan Pedro Toni', 'context': "The Christian Brothers of Ireland Stella Maris College is a private, co-educational, not-for-profit Catholic school located in the wealthy residential southeastern neighbourhood of Carrasco. Established in 1955, it is regarded as one of the best high schools in the country, blending a rigorous curriculum with strong extracurricular activities. The school's headmaster, history professor Juan Pedro Toni, is a member of the Stella Maris Board of Governors and the school is a member of the International Baccalaureate Organization (IBO). Its long list of distinguished former pupils includes economists, engineers, architects, lawyers, politicians and even F1 champions. The school has also played an important part in the development of rugby union in Uruguay, with the creation of Old Christians Club, the school's alumni club.", 'rag_contexts': ["The Christian Brothers of Ireland Stella Maris College is a private, co-educational, not-for-profit Catholic school located in the wealthy residential southeastern neighbourhood of Carrasco. Established in 1955, it is regarded as one of the best high schools in the country, blending a rigorous curriculum with strong extracurricular activities. The school's headmaster, history professor Juan Pedro Toni, is a member of the Stella Maris Board of Governors and the school is a member of the International Baccalaureate Organization (IBO). Its long list of distinguished former pupils includes economists, engineers, architects, lawyers, politicians and even F1 champions. The school has also played an important part in the development of rugby union in Uruguay, with the creation of Old Christians Club, the school's alumni club.", "Red is one of the most common colors used on national flags. The use of red has similar connotations from country to country: the blood, sacrifice, and courage of those who defended their country; the sun and the hope and warmth it brings; and the sacrifice of Christ's blood (in some historically Christian nations) are a few examples. Red is the color of the flags of several countries that once belonged to the former British Empire. The British flag bears the colors red, white, and blue; it includes the cross of Saint George, patron saint of England, and the saltire of Saint Patrick, patron saint of Ireland, both of which are red on white. The flag of the United States bears the colors of Britain, the colors of the French tricolore include red as part of the old Paris coat of arms, and other countries' flags, such as those of Australia, New Zealand, and Fiji, carry a small inset of the British flag in memory of their ties to that country. Many former colonies of Spain, such as Mexico, Colombia, Ecuador, Cuba, Puerto Rico, Peru, and Venezuela, also feature red—one of the colors of the Spanish flag—on their own banners. Red flags are also used to symbolize storms, bad water conditions, and many other dangers. Navy flags are often red and yellow. Red is prominently featured in the flag of the United States Marine Corps."], 'extra': "In July 2015, Eton accidentally sent emails to 400 prospective students, offering them conditional entrance to the school in September 2017. The email was intended for nine students, but an IT glitch caused the email to be sent to 400 additional families, who didn't necessarily have a place. In response, the school issued the following statement: \"This error was discovered within minutes and each family was immediately contacted to notify them that it should be disregarded and to apologise. We take this type of incident very seriously indeed and so a thorough investigation, overseen by the headmaster Tony Little and led by the tutor for admissions, is being carried out to find out exactly what went wrong and ensure it cannot happen again. Eton College offers its sincere apologies to those boys concerned and their families. We deeply regret the confusion and upset this must have caused.\", 'John Evans, for whom Evanston is named, bought 379 acres (153 ha) of land along Lake Michigan in 1853, and Philo Judson developed plans for what would become the city of Evanston, Illinois. The first building, Old College, opened on November 5, 1855. To raise funds for its construction, Northwestern sold $100 \"perpetual scholarships\" entitling the purchaser and his heirs to free tuition. Another building, University Hall, was built in 1869 of the same Joliet limestone as the Chicago Water Tower, also built in 1869, one of the few buildings in the heart of Chicago to survive the Great Chicago Fire of 1871. In 1873 the Evanston College for Ladies merged with Northwestern, and Frances Willard, who later gained fame as a suffragette and as one of the founders of the Woman's Christian Temperance Union (WCTU), became the school's first dean of women. Willard Residential College (1938) is named in her honor. Northwestern admitted its first women students in 1869, and the first woman was graduated in 1874.'"}]
```

'Two years later, the Emperor Valens, who favored the Arian position, in his turn exiled Athanasius. This time however, Athanasius simply left for the outskirts of Alexandria, where he stayed for only a few months before the local authorities convinced Valens to retract his order of exile. Some early reports state that Athanasius spent this period of exile at his family\'s ancestral tomb in a Christian cemetery. It was during this period, the final exile, that he is said to have spent four months in hiding in his father\'s tomb. (Soz., "Hist. Eccl.", VI, xii; Soc., "Hist. Eccl.", IV, xii).']}

Before we run an end-to-end evaluation, we can check the accuracy of the word overlap retriever. In other words, for what fraction of questions is the gold context included in the top-k retrieved contexts.

```
# evaluation metric of retriever
def evaluate_retriever(rag_qa_pairs):
    """
    Evaluates the retriever by computing the accuracy of retrieved contexts against reference contexts.
    """
    correct_retrievals = 0
    for qa_item in rag_qa_pairs:
        if qa_item['context'] in qa_item['rag_contexts']:
            correct_retrievals += 1
    accuracy = correct_retrievals / len(rag_qa_pairs)
    return accuracy
```

In our implementation, we got an accuracy of 0.372.

```
evaluate_retriever(rag_qa_pairs)
0.52
```

**TODO:** Write a function `rag_qa(qa_item)` that behaves like the `vanilla_qa` and `oracle_qa` functions above. Create a prompt from the question and the top-k retrieved contexts (instead of the gold context you used in `oracle_qa`). You can assume that `qa_item` already contains the 'rag\_contexts' field.

```
def rag_qa(qa_item): # Write this function
    prompt = (
        "As a factual question answering system, you are going to answer question.\n"
        "Answer the question using as few words as possible.\n"
        "Do not include articles (a, an, the), punctuation, or explanations.\n"
        "#If unsure, answer with ''.\n\n"
        f"Context: {qa_item['rag_contexts']}"
        f"Question: {qa_item['question']}\n"
        "Answer:"
    )

    output = pipe(prompt,
                  max_new_tokens=128,
                  do_sample=True, # set to False for greedy decoding below
                  pad_token_id=pipe.tokenizer.eos_token_id)

    answer = output[0]['generated_text'][len(prompt):].strip()
    return answer
```

**TODO:** Like you did for the vanilla and oracle qa system, evaluate the `rag_qa` function and display the results. In our implementation, we got an exact match of 19.6%.

```
rag_overlap_eval = evaluate_qa(rag_qa, rag_qa_pairs)
present_results(rag_overlap_eval)

Evaluating QA instances: 100%|██████████| 250/250 [00:51<00:00,  4.89it/s] Evaluation Results:
Exact Match: 30.00%
F1 Score: 39.24%
ROUGE2 F1: 17.95%
Question: Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?
Gold Answer: professor Juan Pedro Toni
Predicted Answer: Juan Pedro Toni
Exact Match: 0, F1 Score: 0.8571428571428571
ROUGE-2 F1-score: 0.8
-----
Question: What is the ratio of black and Asian schoolchildren to white schoolchildren?
Gold Answer: about six to four
Predicted Answer: 10.9%
Exact Match: 0, F1 Score: 0.0
ROUGE-2 F1-score: 0.0
```

```
Question: When did Outcault's The Yellow Kid appear in newspapers?
```

```
Gold Answer: 1890s
```

```
Predicted Answer: 1890s
```

```
Exact Match: 1, F1 Score: 1.0
```

```
ROUGE-2 F1-score: 0.0
```

```
-----
```

```
Question: When did devolution in the UK begin?
```

```
Gold Answer: 1914
```

```
Predicted Answer: 1908
```

```
Exact Match: 0, F1 Score: 0.0
```

```
ROUGE-2 F1-score: 0.0
```

```
-----
```

```
Question: Treating the mitrailleuse like what rendered it far less effective
```

```
Gold Answer: artillery
```

```
Predicted Answer: an ambush
```

```
Exact Match: 0, F1 Score: 0.0
```

```
ROUGE-2 F1-score: 0.0
```

## ▼ Part 5 - Retrieval-Augmented Question Answering - Dense Retrieval

In this step, we will try to encode each context and questions using BERT. We will then retrieve the k contexts whose embeddings have the highest cosine similarity to the question embedding.

### 5.1 Creating Embeddings for Contexts and Questions

Here is an example for how to use BERT to encode a sentence. Instead of using the CLS embeddings (as discussed in class) we will pool together the token representations at the last layer by averaging. The resulting representation is a (1,768) tensor.

```
device = "cuda"
from transformers import BertTokenizer, BertModel # If you run into memory issues, you

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased').to(device)

inputs = tokenizer("This is a sample sentence.", return_tensors="pt", padding=True, truncation=True, max_length=512)
with torch.no_grad():
    outputs = model(**inputs)
    hidden_states = outputs.last_hidden_state
    embedding = torch.mean(hidden_states, dim=1) # (batch_size=1, embedding size =768)
```

```
embedding.shape
```

```
torch.Size([1, 768])
```

**TODO:** Write code to encode each candidate context. Stack the embeddings together into a single (19035, 768) pytorch tensor that we can save to disk and reload as needed (see above for how to access the candidate contexts). On some lower-resource systems you may have trouble instantiating both BERT and OLMo2 at the same time. Storing the encoded representations allows you to run just OLMo for the QA part.

```
batch_size = 32
max_length = 256

embedding_list = []
model.eval()

with torch.no_grad():
    for i in tqdm.tqdm(
        range(0, len(candidate_contexts), batch_size),
        desc="Encoding candidate contexts"
    ):
        batch_contexts = candidate_contexts[i:i + batch_size]

        inputs = tokenizer(
            batch_contexts,
            return_tensors="pt",
            padding=True,
            truncation=True,
            max_length=max_length
        ).to(device)
```

```

outputs = model(**inputs) # (B, L, 768)

# ---- mean pooling ----
attention_mask = inputs["attention_mask"].unsqueeze(-1) # (B, L, 1)
summed = (outputs.last_hidden_state * attention_mask).sum(dim=1)
counts = attention_mask.sum(dim=1).clamp(min=1e-9)
batch_embeddings = summed / counts # (B, 768)

embedding_list.append(batch_embeddings.cpu())

# Stack into a single tensor
context_embeddings = torch.cat(embedding_list, dim=0)
print("Final shape:", context_embeddings.shape)

Encoding candidate contexts: 100%|██████████| 595/595 [01:39<00:00,  5.97it/s]Final shape: torch.Size([19035, 768])

torch.save(context_embeddings, "context_embeddings.pt")

```

**TODO:** Similarly encode each question and stack the embeddings together into a single (250, 768) pytorch tensor that we can save to disk and reload as needed.

```

batch_size = 32
max_length = 64

question_texts = [qa["question"] for qa in evaluation_benchmark["qas"]]

question_embedding_list = []
model.eval()

with torch.no_grad():
    for i in tqdm.tqdm(
        range(0, len(question_texts), batch_size),
        desc="Encoding questions"
    ):
        batch_questions = question_texts[i:i + batch_size]

        inputs = tokenizer(
            batch_questions,
            return_tensors="pt",
            padding=True,
            truncation=True,
            max_length=max_length
        ).to(device)

        outputs = model(**inputs) # (B, L, 768)

        # ---- mean pooling ----
        attention_mask = inputs["attention_mask"].unsqueeze(-1) # (B, L, 1)
        summed = (outputs.last_hidden_state * attention_mask).sum(dim=1)
        counts = attention_mask.sum(dim=1).clamp(min=1e-9)
        batch_embeddings = summed / counts # (B, 768)

        question_embedding_list.append(batch_embeddings.cpu())

# Stack into (250, 768)
question_embeddings = torch.cat(question_embedding_list, dim=0)

print("Final question embedding shape:", question_embeddings.shape)

Encoding questions: 100%|██████████| 8/8 [00:00<00:00, 45.19it/s]Final question embedding shape: torch.Size([250, 768])

torch.save(question_embeddings, "question_embeddings.pt")

```

## ▼ 5.2 Similarity Retriever

```

context_embeddings = torch.load("context_embeddings.pt")
question_embeddings = torch.load("question_embeddings.pt")

```

**TODO:** Write a function `retrieve_cosine(question_embedding, contexts, context_embeddings)` that takes in the embedding for a single question (a [1,768] tensor), a list of contexts (each is a string), and the context embedding tensor [19035,768]. Note that the indices of the context list and the rows of the context\_embeddings tensor line up. i.e. `context_embeddings[0]` is the embedding for `contexts[0]`, etc. You can use `torch.nn.functional.cosine_similarity` (or `F.cosine_similarity` since we imported `torch.nn.functional` as `F`, which is conventional) to calculate the similarities efficiently. You may also want to look at `torch.topk`, but other solutions are possible.

```
def retrieve_cosine(question_emb, contexts, context_embeddings, top_k=5):
    """
    question_emb: Tensor of shape (1, 768)
    contexts: list of context strings (length 19035)
    context_embeddings: Tensor of shape (19035, 768)
    """

    # Ensure shapes are compatible
    if question_emb.dim() == 2:
        question_emb = question_emb.squeeze(0)  # (768,)

    # Compute cosine similarities: (19035,)
    similarities = F.cosine_similarity(
        context_embeddings,  # (19035, 768)
        question_emb.unsqueeze(0),  # (1, 768) -> broadcast
        dim=1
    )

    # indices of top-k most similar contexts
    topk_scores, topk_indices = torch.topk(similarities, k=top_k)

    # Return contexts
    return [contexts[i] for i in topk_indices.tolist()]
```

```
retrieve_cosine(question_embeddings[0], candidate_contexts, context_embeddings)
```

"The Christian Brothers of Ireland Stella Maris College is a private, co-educational, not-for-profit Catholic school located in the wealthy residential southeastern neighbourhood of Carrasco. Established in 1955, it is regarded as one of the best high schools in the country, blending a rigorous curriculum with strong extracurricular activities. The school's headmaster, history professor Juan Pedro Toni, is a member of the Stella Maris Board of Governors and the school is a member of the International Baccalaureate Organization (IBO). Its long list of distinguished former pupils includes economists, engineers, architects, lawyers, politicians and even F1 champions. The school has also played an important part in the development of rugby union in Uruguay, with the creation of Old Christians Club, the school's alumni club.",

'The National Maritime College of Ireland is also located in Cork and is the only college in Ireland in which Nautical Studies and Marine Engineering can be undertaken. CIT also incorporates the Cork School of Music and Crawford College of Art and Design as constituent schools. The Cork College of Commerce is the largest post-Leaving Certificate college in Ireland and is also the biggest provider of Vocational Preparation and Training courses in the country. [citation needed] Other 3rd level institutions include Griffith College Cork, a private institution, and various other colleges.',

'The University of St Mark & St John (known as "Marjon" or "Marjons") specialises in teacher training, and offers training across the country and abroad.',

"Eton College has links with some private schools in India today, maintained from the days of the British Raj, such as The Doon School and Mayo College. Eton College is also a member of the G20 Schools Group, a collection of college preparatory boarding schools from around the world, including Turkey's Robert College, the United States' Phillips Academy and Phillips Exeter Academy, Australia's Scotch College, Melbourne Grammar School and Launceston Church Grammar School, Singapore's Raffles Institution, and Switzerland's International School of Geneva. Eton has recently fostered [when?] a relationship with the Roxbury Latin School, a traditional all-boys private school in Boston, USA. Former Eton headmaster and provost Sir Eric Anderson shares a close friendship with Roxbury Latin Headmaster emeritus F. Washington Jarvis; Anderson has visited Roxbury Latin on numerous occasions, while Jarvis briefly taught theology at Eton after retiring from his headmaster post at Roxbury Latin. The headmasters' close friendship spawned the Hennessy Scholarship, an annual prize established in 2005 and awarded to a graduating RL senior for a year of study at Eton. Hennessy Scholars generally reside in Wotton house.",

"Detroit is served by various private schools, as well as parochial Roman Catholic schools operated by the Archdiocese of Detroit. As of 2013 [update] there are four Catholic grade schools and three Catholic high schools in the City of Detroit, with all of them in the city's west side. The Archdiocese of Detroit lists a number of primary and secondary schools in the metro area as Catholic education has emigrated to the suburbs. Of the three Catholic high schools in the city, two are operated by the Society of Jesus and the third is co-sponsored by the Sisters, Servants of the Immaculate Heart of Mary and the Congregation of St. Basil."]

**TODO:** Write a new version of the `add_rag_context` function we provided above. This function should now additionally take the question embeddings and context embeddings as parameters, run the retrieval for each question (using the `retrieve_cosine` function above) and populate a new list of `qa_items`, include the selected 'rag\_contexts'.

```
def add_rag_context_dense(qa_items, contexts, retriever, question_embeddings, context_embeddings, top_k=5):
    """
    qa_items: list of QA dicts (length 250)
```

```
contexts: list of all candidate contexts (length 19035)
retriever: retrieve_cosine function
question_embeddings: Tensor (250, 768)
context_embeddings: Tensor (19035, 768)
"""
```

```
result_items = copy.deepcopy(qa_items)

for i, qa_item in tqdm.tqdm(
    enumerate(result_items),
    total=len(result_items),
    desc="Retrieving dense contexts"
):
    question_emb = question_embeddings[i].unsqueeze(0) # (1, 768)

    retrieved_contexts = retriever(
        question_emb,
        contexts,
        context_embeddings,
        top_k=top_k
    )

    qa_item["rag_contexts"] = retrieved_contexts

return result_items
```

```
rag_qa_items = add_rag_context_dense(evaluation_benchmark['qas'], candidate_contexts, retrieve_cosine, question_embe
```

```
Retrieving dense contexts: 100%|██████████| 250/250 [00:07<00:00, 31.62it/s]
```

```
rag_qa_items[0]
```

```
{"question": "Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?", "answer": "professor Juan Pedro Toni", "context": "The Christian Brothers of Ireland Stella Maris College is a private, co-educational, not-for-profit Catholic school located in the wealthy residential southeastern neighbourhood of Carrasco. Established in 1955, it is regarded as one of the best high schools in the country, blending a rigorous curriculum with strong extracurricular activities. The school's headmaster, history professor Juan Pedro Toni, is a member of the Stella Maris Board of Governors and the school is a member of the International Baccalaureate Organization (IBO). Its long list of distinguished former pupils includes economists, engineers, architects, lawyers, politicians and even F1 champions. The school has also played an important part in the development of rugby union in Uruguay, with the creation of Old Christians Club, the school's alumni club.", "rag_contexts": ["The Christian Brothers of Ireland Stella Maris College is a private, co-educational, not-for-profit Catholic school located in the wealthy residential southeastern neighbourhood of Carrasco. Established in 1955, it is regarded as one of the best high schools in the country, blending a rigorous curriculum with strong extracurricular activities. The school's headmaster, history professor Juan Pedro Toni, is a member of the Stella Maris Board of Governors and the school is a member of the International Baccalaureate Organization (IBO). Its long list of distinguished former pupils includes economists, engineers, architects, lawyers, politicians and even F1 champions. The school has also played an important part in the development of rugby union in Uruguay, with the creation of Old Christians Club, the school's alumni club.", "The National Maritime College of Ireland is also located in Cork and is the only college in Ireland in which Nautical Studies and Marine Engineering can be undertaken. CIT also incorporates the Cork School of Music and Crawford College of Art and Design as constituent schools. The Cork College of Commerce is the largest post-Leaving Certificate college in Ireland and is also the biggest provider of Vocational Preparation and Training courses in the country.[citation needed] Other 3rd level institutions include Griffith College Cork, a private institution, and various other colleges.", "The University of St Mark & St John (known as "Marjon" or "Marjons") specialises in teacher training, and offers training across the country and abroad.", "Eton College has links with some private schools in India today, maintained from the days of the British Raj, such as The Doon School and Mayo College. Eton College is also a member of the G20 Schools Group, a collection of college preparatory boarding schools from around the world, including Turkey's Robert College, the United States' Phillips Academy and Phillips Exeter Academy, Australia's Scotch College, Melbourne Grammar School and Launceston Church Grammar School, Singapore's Raffles Institution, and Switzerland's International School of Geneva. Eton has recently fostered[when?] a relationship with the Roxbury Latin School, a traditional all-boys private school in Boston, USA. Former Eton headmaster and provost Sir Eric Anderson shares a close friendship with Roxbury Latin Headmaster emeritus F. Washington Jarvis; Anderson has visited Roxbury Latin on numerous occasions, while Jarvis briefly taught theology at Eton after retiring from his headmaster post at Roxbury Latin. The headmasters' close friendship spawned the Hennessy Scholarship, an annual prize established in 2005 and awarded to a graduating RL senior for a year of study at Eton. Hennessy Scholars generally reside in Wotton house.", "Detroit is served by various private schools, as well as parochial Roman Catholic schools operated by the Archdiocese of Detroit. As of 2013[update] there are four Catholic grade schools and three Catholic high schools in the City of Detroit, with all of them in the city's west side. The Archdiocese of Detroit lists a number of primary and secondary schools in the metro area as Catholic education has emigrated to the suburbs. Of the three Catholic high schools in the city, two are operated by the Society of Jesus and the third is co-sponsored by the Sisters, Servants of the Immaculate Heart of Mary and the Congregation of St. Basil."]}
```

Run the `evaluate_retriever` function on the new `qa_items`. In our experiments, we got an accuracy of about 0.4.

```
evaluate_retriever(rag_qa_items)
0.412
```

Then, evaluate the rag\_qa approach using the revised rag\_qa\_items. You should get an Exact match better than 20%.

```
result = evaluate_qa(rag_qa, rag_qa_items)
present_results(result)

Evaluating QA instances: 100%|██████████| 250/250 [00:35<00:00,  6.96it/s] Evaluation Results:
Exact Match: 24.00%
F1 Score: 32.98%
ROUGE2 F1: 13.64%
Question: Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?
Gold Answer: professor Juan Pedro Toni
Predicted Answer: Juan Pedro Toni
Exact Match: 0, F1 Score: 0.8571428571428571
ROUGE-2 F1-score: 0.8
-----
Question: What is the ratio of black and Asian schoolchildren to white schoolchildren?
Gold Answer: about six to four
Predicted Answer: 1
Exact Match: 0, F1 Score: 0.0
ROUGE-2 F1-score: 0.0
-----
Question: When did Outcault's The Yellow Kid appear in newspapers?
Gold Answer: 1890s
Predicted Answer: 1896
Exact Match: 0, F1 Score: 0.0
ROUGE-2 F1-score: 0.0
-----
Question: When did devolution in the UK begin?
Gold Answer: 1914
Predicted Answer: 1914
Exact Match: 1, F1 Score: 1.0
ROUGE-2 F1-score: 0.0
-----
Question: Treating the mitrailleuse like what rendered it far less effective
Gold Answer: artillery
Predicted Answer: French
Exact Match: 0, F1 Score: 0.0
ROUGE-2 F1-score: 0.0
```

## ▼ Part 6 - Experiments

**TODO** For the overlap and dense retrievers (from part 5 and 6), what happens when you change the number of retrieved contexts? Present a table of results for k=1, k=5 (already done), k=10, and k=20.

```
ks = [1, 5, 10, 20]

overlap_results = {}

for k in ks:
    print(f"\nOverlap RAG k={k}")
    rag_pairs = add_rag_context_overlap(
        evaluation_benchmark["qas"],
        candidate_contexts,
        retrieve_overlap,
        top_k=k
    )
    eval_results = evaluate_qa(rag_qa, rag_pairs)
    present_results(eval_results, f"Overlap RAG (k={k})")

for k in ks:
    print(f"\nDense RAG k={k}")
    rag_pairs = add_rag_context_dense(
        evaluation_benchmark["qas"],
        candidate_contexts,
        retrieve_cosine,
        question_embeddings,
        context_embeddings,
        top_k=k
    )
```

```
eval_results = evaluate_qa(rag_qa, rag_pairs)
present_results(eval_results, f"Dense RAG (k={k})")
```

Question: When did Outcault's The Yellow Kid appear in newspapers?

Gold Answer: 1890s

Predicted Answer: 1890s

Exact Match: 1, F1 Score: 1.0

ROUGE-2 F1-score: 0.0

-----

Question: When did devolution in the UK begin?

Gold Answer: 1914

Predicted Answer: 1908

Exact Match: 0, F1 Score: 0.0

ROUGE-2 F1-score: 0.0

-----

Question: Treating the mitrailleuse like what rendered it far less effective

Gold Answer: artillery

Predicted Answer: no

Exact Match: 0, F1 Score: 0.0

ROUGE-2 F1-score: 0.0

-----

Overlap RAG k=10

Retrieving contexts: 100%|██████████| 250/250 [08:21<00:00, 2.01s/it]

Evaluating QA instances: 28%|██| 70/250 [00:14<00:33, 5.38it/s] This is a friendly reminder – the current i

Evaluating QA instances: 100%|██████████| 250/250 [01:00<00:00, 4.14it/s]

Overlap RAG (k=10) Evaluation Results:

Exact Match: 27.20%

F1 Score: 38.02%

ROUGE2 F1: 18.05%

Question: Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?

Gold Answer: professor Juan Pedro Toni

Predicted Answer: Juan Pedro Toni

Exact Match: 0, F1 Score: 0.8571428571428571

ROUGE-2 F1-score: 0.8

-----

Question: What is the ratio of black and Asian schoolchildren to white schoolchildren?

Gold Answer: about six to four

Predicted Answer: 3.4%

Exact Match: 0, F1 Score: 0.0

ROUGE-2 F1-score: 0.0

-----

Question: When did Outcault's The Yellow Kid appear in newspapers?

Gold Answer: 1890s

Predicted Answer: 1890s

Exact Match: 1, F1 Score: 1.0

ROUGE-2 F1-score: 0.0

-----

Question: When did devolution in the UK begin?

Gold Answer: 1914

Predicted Answer: 1973

Exact Match: 0, F1 Score: 0.0

ROUGE-2 F1-score: 0.0

-----

Question: Treating the mitrailleuse like what rendered it far less effective

Gold Answer: artillery

Predicted Answer: less reflective and thus less effective

Exact Match: 0, F1 Score: 0.0

ROUGE-2 F1-score: 0.0

-----

```
import pandas as pd
```

```
data = [
```

```
    ["Overlap", 1, 21.20, 28.95, 10.39],
    ["Overlap", 5, 32.40, 40.29, 18.25],
    ["Overlap", 10, 27.20, 38.02, 18.05],
    ["Overlap", 20, 10.00, 15.23, 7.18],
    ["Dense", 1, 14.40, 19.21, 5.60],
    ["Dense", 5, 24.40, 32.92, 13.43],
    ["Dense", 10, 28.40, 36.62, 17.58],
    ["Dense", 20, 20.00, 31.87, 13.54],
```

```
]
```

```
df = pd.DataFrame(
```

```
    data,
```

```
    columns=["Retriever", "k", "Exact Match (%)", "F1 Score (%)", "ROUGE-2 F1 (%)"]
```

```
)
```

```
df
```

	Retriever	k	Exact Match (%)	F1 Score (%)	ROUGE-2 F1 (%)	
0	Overlap	1	21.2	28.95	10.39	
1	Overlap	5	32.4	40.29	18.25	
2	Overlap	10	27.2	38.02	18.05	
3	Overlap	20	10.0	15.23	7.18	
4	Dense	1	14.4	19.21	5.60	
5	Dense	5	24.4	32.92	13.43	
6	Dense	10	28.4	36.62	17.58	
7	Dense	20	20.0	31.87	13.54	

Next steps: [Generate code with df](#) [New interactive sheet](#)

Double-click (or enter) to edit

## ▼ Part 7 -Improving the QA System

**TODO** In this part, we ask you to come up with one interesting or novel idea for improving the QA system. Your system does *not* have to outperform the models from part 4 or 5, but for full credit you should implement at least one new idea, beyond just changing parameters. You can either work on better retrieval or better QA/LLM performance. Show the full code for the necessary steps and evaluation results.

Ideas for improving the retriever include: improved word overlap (better tokenization/ text normalization, using TF-IDF, ...), or choosing a different approach or different model (other than BERT) for calculating context and question embeddings.

For the LLM, you could try a different transformer model, including text-to-text models (e.g. T5).

```
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

# build TF-IDF index over candidate contexts
tfidf_vectorizer = TfidfVectorizer(
    tokenizer=get_tokens,
    lowercase=False,
    preprocessor=None,
    token_pattern=None
)

tfidf_matrix = tfidf_vectorizer.fit_transform(candidate_contexts)

def retrieve_tfidf(question, contexts, top_k=5):
    q_vec = tfidf_vectorizer.transform([question]) # (1, vocab)

    # cosine similarity via dot product (TF-IDF vectors are L2-normalized)
    scores = (tfidf_matrix @ q_vec.T).toarray().squeeze()

    top_idx = np.argsort(scores)[::-1][:top_k]
    return [contexts[i] for i in top_idx]

#top k 8 had the highest result
rag_qa_pairs_tfidf = add_rag_context_overlap(evaluation_benchmark['qas'], candidate_contexts, retrieve_tfidf, top_k=8)

evaluate_retriever(rag_qa_pairs_tfidf)
rag_tfidf_eval = evaluate_qa(rag_qa, rag_qa_pairs_tfidf)
present_results(rag_tfidf_eval)

Retrieving contexts: 100%|██████████| 250/250 [00:01<00:00, 139.67it/s]
Evaluating QA instances: 100%|██████████| 250/250 [00:56<00:00,  4.42it/s] Evaluation Results:
Exact Match: 36.40%
F1 Score: 46.67%
ROUGE2 F1: 23.14%
Question: Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?
Gold Answer: professor Juan Pedro Toni
Predicted Answer: Juan Pedro Toni
Exact Match: 0, F1 Score: 0.8571428571428571
ROUGE-2 F1-score: 0.8

Question: What is the ratio of black and Asian schoolchildren to white schoolchildren?
Gold Answer: about six to four
```

Predicted Answer: 1/2  
Exact Match: 0, F1 Score: 0.0  
ROUGE-2 F1-score: 0.0

---

Question: When did Outcault's The Yellow Kid appear in newspapers?

Gold Answer: 1890s

Predicted Answer: 1890s

Exact Match: 1, F1 Score: 1.0

ROUGE-2 F1-score: 0.0

---

Question: When did devolution in the UK begin?

Gold Answer: 1914

Predicted Answer: devolution in the UK began with the Government of Ireland Act 1914

Exact Match: 0, F1 Score: 0.181818181818182

ROUGE-2 F1-score: 0.0

---

Question: Treating the mitrailleuse like what rendered it far less effective

Gold Answer: artillery

Predicted Answer: artillery

Exact Match: 1, F1 Score: 1.0

ROUGE-2 F1-score: 0.0

---