COMS W4705 - Homework 4

Question Answering with Retrieval Augmented Generation

Anubhav Jangra <aj3228@columbia.edu>, Emile Al-Billeh <ea3048@columbia.edu>, Daniel Bauer <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are lucky and the LLM has have enough information to answer this question from pre-training, but the information may be outdated (the headmaster is likely to change from time to time).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 (4.57.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from transformers) (3.2
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from transformers) (
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from transformer
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.12/dist-packages (from transformers) (
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.12/dist-packages (from transform
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from transformers) (2.3
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from transfor
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.12/dist-packages (from transformers) (4
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.12/dist-packages (from huggingfac
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.12/dist-packages (from 
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggin
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from re
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->tran
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests
!pip install accelerate
```

```
Requirement already satisfied: accelerate in /usr/local/lib/python3.12/dist-packages (1.11.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.12/dist-packages (from accelerate) (2.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from accelerate)
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
Requirement already satisfied: huggingface_hub>=0.21.0 in /usr/local/lib/python3.12/dist-packages (from acc
Requirement already satisfied: safetensors>=0.4.3 in /usr/local/lib/python3.12/dist-packages (from accelera
Requirement already satisfied: filelock in /usr/local/lib/python3.12/dist-packages (from huggingface_hub>=0
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.12/dist-packages (from huggingfac
Requirement already satisfied: requests in /usr/local/lib/python3.12/dist-packages (from huggingface_hub>=0
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.12/dist-packages (from huggingface_hu
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in /usr/local/lib/python3.12/dist-packages (from huggin
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->ac
Requirement already satisfied: sympy>=1.13.3 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
Requirement already satisfied: networkx>=2.5.1 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0->accele
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in /usr/local/lib/python3.12/dist-packages (fr
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in /usr/local/lib/python3.12/dist-packages (fr
Requirement already satisfied: nvidia-nccl-cu12==2.27.5 in /usr/local/lib/python3.12/dist-packages (from to
Requirement already satisfied: nvidia-nvshmem-cu12==3.3.20 in /usr/local/lib/python3.12/dist-packages (from
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in /usr/local/lib/python3.12/dist-packages (from t
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in /usr/local/lib/python3.12/dist-packages (f
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in /usr/local/lib/python3.12/dist-packages (fro
Requirement already satisfied: triton==3.5.0 in /usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.12/dist-packages (from sympy>=1
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->tor
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from re
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests->hugg
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests
```

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'][:5]:
    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 (/biːˈjɒnseɪ/ bee-YON-say) (born September 4, 1981) is an American singer, s
==============================
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
--------------------
```

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

```
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 tas
# 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 i
      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_coun
print(f"Unanswerable questions: {unanswerable_count} ({unanswerable_count / (answerable_count + unanswerab
```

```
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 ha
--------------------
Two years later, the Emperor Valens, who favored the Arian position, in his turn exiled Athanasius. This tin
--------------------
==============================
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 Cathol
ic school located in the wealthy residential southeastern neighbourhood of Carrasco. Established in 1955, i
t is regarded as one of the best high schools in the country, blending a rigorous curriculum with strong ex
tracurricular activities. The school's headmaster, history professor Juan Pedro Toni, is a member of the St
ella Maris Board of Governors and the school is a member of the International Baccalaureate Organization (I
BO). Its long list of distinguished former pupils includes economists, engineers, architects, lawyers, poli
ticians and even F1 champions. The school has also played an important part in the development of rugby uni
on in Uruguay, with the creation of Old Christians Club, the school's alumni club.'
```

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 leve, 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):
    gold_toks = get_tokens(a_gold)
    pred_toks = get_tokens(a_pred)

    if len(gold_toks) == 0 or len(pred_toks) == 0:
        return int(gold_toks == pred_toks)

    common = collections.Counter(gold_toks) & collections.Counter(pred_toks)
    num_same = sum(common.values())

    if num_same == 0:
        return 0

    precision = num_same / len(pred_toks)
    recall = num_same / len(gold_toks)
    f1 = (2 * precision * recall) / (precision + recall)

    return f1
```

```
# Test your function
test_gold = "The capital of England is London."
test_pred = "London, capital of England"

print(f"Gold: {test_gold}")
print(f"Pred: {test_pred}")
print(f"F1 Score: {compute_f1(test_gold, test_pred)}")

# Test edge cases
print(f"\nEdge case - identical: {compute_f1('London', 'London')}")
print(f"Edge case - no overlap: {compute_f1('Paris', 'London')}")
print(f"Edge case - empty: {compute_f1('', '')}")
```

```
Gold: The capital of England is London.
Pred: London, capital of England
F1 Score: 0.888888888888889

Edge case - identical: 1.0
Edge case - no overlap: 0
Edge case - empty: 1
```

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
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk->rouge_score) (8
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk->rouge_score) (
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk->rouge_
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk->rouge_score) (4.
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=344c1563c715
  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: 0.888888888888889
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/sett
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, 14.2kB/s]
model.safetensors: 100%                                    2.97G/2.97G [00:36<00:00, 119MB/s]
generation_config.json: 100%                               121/121 [00:00<00:00, 13.4kB/s]
tokenizer_config.json:     4.88k/? [00:00<00:00, 461kB/s]
vocab.json:     1.61M/? [00:00<00:00, 4.38MB/s]
merges.txt:     917k/? [00:00<00:00, 20.6MB/s]
tokenizer.json:     7.14M/? [00:00<00:00, 84.2MB/s]
special_tokens_map.json: 100%                              581/581 [00:00<00:00, 67.4kB/s]
Device set to use cuda
```

We can now pass a prompt to the model and retreive 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 to make homemade apple sauce, for use during the wi
```

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

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

```
'to make homemade apple sauce, for use during the winter months when fresh fruit is scarce. The process is
simple: gather apples, cut them into chunks, and simmer them down until they're soft and sweet. The result
is a warm, comforting dish that's perfect for eating with a spoon or as a topping for pancakes or ice crea
m. \n\nHere's a step-by-step guide on how to make homemade apple sauce:\n\nIngredients:\n* 4 pounds ripe, s
weet apples (Granny Smith, Fuji, Braeburn, etc.)\n* Water or apple juice (about 4 cups)\n* Sugar (to taste)
\n* A bit of'
```

Using the LLM for Question Answering

TODO: Write a function `vanilla_qa(qa_item)` that take 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 he 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):
    question = qa_item['question']

    # Create a simple prompt with an example (single-shot)
    prompt = """Answer the following question concisely with just a few words or a short phrase.

Example:
Question: Who wrote the novel "Pride and Prejudice"?
Answer: Jane Austen

Question: """ + question + """
Answer:"""
```

```
    output = pipe(prompt,
                  max_new_tokens=50,
                  do_sample=False,  # Use greedy decoding for consistency
                  pad_token_id=pipe.tokenizer.eos_token_id)

    # Extract just the answer, removing the prompt
    answer = output[0]['generated_text'][len(prompt):].strip()

    # Clean up the answer - take only the first line or sentence
    answer = answer.split('\n')[0].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']
```

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

```
vanilla_qa(qa_item) # inspect the item
```

```
The following generation flags are not valid and may be ignored: ['temperature']. Set `TRANSFORMERS_VERBOSI
'Headmaster'
```

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: {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:   3%|▌          | 8/250 [00:01<00:58,  4.11it/s]You seem to be using the pipelines 
Evaluating QA instances: 100%|██████████| 250/250 [00:48<00:00,  5.20it/s]
```

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

```
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': 'Headmaster',
 '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,
 'rouge2_f1': 0.0}
```

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: 7.60%
F1 Score: 12.31%
ROUGE2 F1: 3.96%
Question: Who is the headmaster of the Christian Brothers of Ireland Stella Maris College?
Gold Answer: professor Juan Pedro Toni
Predicted Answer: Headmaster
Exact Match: 0, F1 Score: 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: Asian:black ratio
Exact Match: 0, F1 Score: 0
ROUGE-2 F1-score: 0.0
----------------------------------------
Question: When did Outcault's The Yellow Kid appear in newspapers?
Gold Answer: 1890s
Predicted Answer: 1893
Exact Match: 0, F1 Score: 0
ROUGE-2 F1-score: 0.0
----------------------------------------
Question: When did devolution in the UK begin?
Gold Answer: 1914
Predicted Answer: 1997
Exact Match: 0, F1 Score: 0
ROUGE-2 F1-score: 0.0
----------------------------------------
Question: Treating the mitrailleuse like what rendered it far less effective
Gold Answer: artillery
Predicted Answer: less effective
Exact Match: 0, F1 Score: 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):
    question = qa_item['question']
    context = qa_item['context']

    # Create a prompt with context and question
    prompt = f"""Use the following context to answer the question concisely.

Context: {context}

Question: {question}
Answer:"""

    output = pipe(prompt,
                  max_new_tokens=50,
                  do_sample=False,  # Use greedy decoding for consistency
                  pad_token_id=pipe.tokenizer.eos_token_id)
    # Extract just the answer, removing the prompt
    answer = output[0]['generated_text'][len(prompt):].strip()

    # Clean up the answer - take only the first line
    answer = answer.split('\n')[0].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, "Oracle QA")
```

```
Evaluating QA instances: 100%|██████████| 250/250 [02:57<00:00,  1.41it/s]Oracle QA Evaluation Results:
Exact Match: 34.80%
F1 Score: 57.04%
ROUGE2 F1: 30.36%
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 is the headmaster of the Christian Brothers of Ireland Stella Maris Colle
Exact Match: 0, F1 Score: 0.3529411764705882
ROUGE-2 F1-score: 0.23529411764705882
----------------------------------------
Question: What is the ratio of black and Asian schoolchildren to white schoolchildren?
Gold Answer: about six to four
Predicted Answer: Black and Asian children outnumber White British children by about six to four in state s
Exact Match: 0, F1 Score: 0.4
ROUGE-2 F1-score: 0.33333333333333337
----------------------------------------
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: The United Kingdom has traditionally been governed as a unitary state by the Westminster 
Exact Match: 0, F1 Score: 0
ROUGE-2 F1-score: 0.0
----------------------------------------
Question: Treating the mitrailleuse like what rendered it far less effective
Gold Answer: artillery
Predicted Answer: French gunners treated the mitrailleuse like artillery and in this role it was ineffectiv
Exact Match: 0, F1 Score: 0.14285714285714288
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 h
ome 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. Ot
her projects at the development stage include Sangtuda-2 by Iran, Zerafshan by the Chinese company SinoHydr
o, and the Rogun power plant that, at a projected height of 335 metres (1,099 ft), would supersede the Nure
k Dam as highest in the world if it is brought to completion. A planned project, CASA 1000, will transmit 1
000 MW of surplus electricity from Tajikistan to Pakistan with power transit through Afghanistan. The total
length of transmission line is 750 km while the project is planned to be on Public-Private Partnership basi
s with the support of WB, IFC, ADB an...'
```

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)` 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
def retrieve_overlap(question, contexts, top_k=5):
    question_tokens = set(get_tokens(question))
    # Calculate overlap score for each context
    overlap_scores = []
    for context in contexts:
        context_tokens = set(get_tokens(context))
        overlap = len(question_tokens & context_tokens)
        overlap_scores.append(overlap)

    # Get indices of top_k contexts with highest overlap
    top_k_indices = sorted(range(len(overlap_scores)),
                          key=lambda i: overlap_scores[i],
                          reverse=True)[:top_k]

    # Return the top_k contexts
    return [contexts[i] for i in top_k_indices]
```

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(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(evaluation_benchmark['qas'], candidate_contexts, retrieve_overlap)
```

```
Retrieving contexts: 100%|██████████| 250/250 [08:21<00:00,  2.00s/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.",
  '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):
    question = qa_item['question']
    rag_contexts = qa_item['rag_contexts']

    # Combine all retrieved contexts
    combined_context = "\n\n".join(rag_contexts)

    # Create a prompt with retrieved contexts and question
    prompt = f"""Use the following contexts to answer the question concisely.

Contexts:
{combined_context}

Question: {question}
Answer:"""

    output = pipe(prompt,
                  max_new_tokens=50,
                  do_sample=False,  # Use greedy decoding for consistency
                  pad_token_id=pipe.tokenizer.eos_token_id)

    # Extract just the answer, removing the prompt
    answer = output[0]['generated_text'][len(prompt):].strip()

    # Clean up the answer - take only the first line
    answer = answer.split('\n')[0].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, "RAG QA with Overlap Retrieval")
```

```
Evaluating QA instances: 100%|██████████| 250/250 [06:53<00:00,  1.65s/it]RAG QA with Overlap Retrieval Eva
Exact Match: 24.80%
F1 Score: 37.71%
ROUGE2 F1: 18.69%
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: The question does not provide enough context to answer the question accurately. The answe
Exact Match: 0, F1 Score: 0.09523809523809523
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: 1895
Exact Match: 0, F1 Score: 0
ROUGE-2 F1-score: 0.0
----------------------------------------
```

```
Question: Treating the mitrailleuse like what rendered it far less effective
Gold Answer: artillery
Predicted Answer: The use of the mitrailleuse, a rapid-fire weapon, was rendered far less effective due to
Exact Match: 0, F1 Score: 0
ROUGE-2 F1-score: 0.0
----------------------------------------
```

Part 5 - Retrieval-Augmented Question Answering - Dense Retrieval

In this step, we will try to will 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_l
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)
```

```
tokenizer_config.json: 100%                                48.0/48.0 [00:00<00:00, 5.23kB/s]
vocab.txt: 100%                                232k/232k [00:00<00:00, 1.83MB/s]
tokenizer.json: 100%                                466k/466k [00:00<00:00, 3.68MB/s]
config.json: 100%                                570/570 [00:00<00:00, 69.3kB/s]
model.safetensors: 100%                                440M/440M [00:04<00:00, 135MB/s]
```

```
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.

```
embedding_list = []

with torch.no_grad():
    # Encode contexts in batches to manage memory
    batch_size = 32
    for i in tqdm.tqdm(range(0, len(candidate_contexts), batch_size), desc="Encoding contexts"):
        batch_contexts = candidate_contexts[i:i+batch_size]

        # Tokenize the batch
        inputs = tokenizer(batch_contexts, return_tensors="pt", padding=True,
                          truncation=True, max_length=512).to(device)

        # Get BERT embeddings
        outputs = model(**inputs)
        hidden_states = outputs.last_hidden_state

        # Average pooling over tokens (batch_size, embedding_size)
        batch_embeddings = torch.mean(hidden_states, dim=1)

        embedding_list.append(batch_embeddings.cpu())

    # Stack all embeddings into a single tensor
    context_embeddings = torch.cat(embedding_list, dim=0)
```

```
Encoding contexts: 100%|██████████| 595/595 [06:23<00:00,  1.55it/s]
```

```
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.

```
embedding_list = []

with torch.no_grad():
    # Encode questions in batches
    batch_size = 32
    questions = [qa['question'] for qa in evaluation_benchmark['qas']]

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

        # Tokenize the batch
        inputs = tokenizer(batch_questions, return_tensors="pt", padding=True,
                          truncation=True, max_length=512).to(device)

        # Get BERT embeddings
        outputs = model(**inputs)
        hidden_states = outputs.last_hidden_state

        # Average pooling over tokens (batch_size, embedding_size)
        batch_embeddings = torch.mean(hidden_states, dim=1)

        embedding_list.append(batch_embeddings.cpu())

    # Stack all embeddings into a single tensor
    question_embeddings = torch.cat(embedding_list, dim=0)
```

```
Encoding questions: 100%|██████████| 8/8 [00:00<00:00, 19.87it/s]
```

```
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 ant to look at `torch.topk`, but other solutions are possible.

```
def retrieve_cosine(question_emb, contexts, context_embeddings, top_k=5):
    # Ensure question_emb is 2D (1, 768)
    if question_emb.dim() == 1:
        question_emb = question_emb.unsqueeze(0)

    # Calculate cosine similarity between question and all contexts
    # F.cosine_similarity expects same dimensions, so we need to expand question_emb
    similarities = F.cosine_similarity(question_emb, context_embeddings, dim=1)

    # Get top_k indices with highest similarity
    top_k_values, top_k_indices = torch.topk(similarities, top_k)

    # Return the top_k contexts
    return [contexts[i] for i in top_k_indices.cpu().numpy()]
```

```
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.",
 "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.",
 '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.',
 "Other Christian denominations on the island include: Roman Catholic (since 1852), Salvation Army (since
1884), Baptist (since 1845) and, in more recent times, Seventh-day Adventist (since 1949), New Apostolic
and Jehovah's Witnesses (of which one in 35 residents is a member, the highest ratio of any country). The
Catholics are pastorally served by the Mission sui iuris of Saint Helena, Ascension and Tristan da Cunha,
whose office of ecclesiastical superior is vested in the Apostolic Prefecture of the Falkland Islands.",
 'The university is the major seat of the Congregation of Holy Cross (albeit not its official headquarters,
which are in Rome). Its main seminary, Moreau Seminary, is located on the campus across St. Joseph lake
from the Main Building. Old College, the oldest building on campus and located near the shore of St. Mary
lake, houses undergraduate seminarians. Retired priests and brothers reside in Fatima House (a former
retreat center), Holy Cross House, as well as Columba Hall near the Grotto. The university through the
Moreau Seminary has ties to theologian Frederick Buechner. While not Catholic, Buechner has praised writers
from Notre Dame and Moreau Seminary created a Buechner Prize for Preaching.']
```

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(qa_items, contexts, retriever, question_embeddings, context_embeddings, top_k=5):
    result_items = copy.deepcopy(qa_items)

    for i, inst in tqdm.tqdm(enumerate(result_items), desc="Retrieving contexts", total=len(result_items))
        question_emb = question_embeddings[i]
        retrieved_contexts = retriever(question_emb, contexts, context_embeddings, top_k)
        inst['rag_contexts'] = retrieved_contexts

    return result_items
```

```
rag_qa_items = add_rag_context(evaluation_benchmark['qas'], candidate_contexts, retrieve_cosine, question_
```

```
Retrieving contexts: 100%|██████████| 250/250 [00:29<00:00,  8.51it/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.",
  "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.",
  '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.',
  "Other Christian denominations on the island include: Roman Catholic (since 1852), Salvation Army (since
1884), Baptist (since 1845) and, in more recent times, Seventh-day Adventist (since 1949), New Apostolic
and Jehovah's Witnesses (of which one in 35 residents is a member, the highest ratio of any country). The
Catholics are pastorally served by the Mission sui iuris of Saint Helena, Ascension and Tristan da Cunha,
whose office of ecclesiastical superior is vested in the Apostolic Prefecture of the Falkland Islands.",
  'The university is the major seat of the Congregation of Holy Cross (albeit not its official
headquarters, which are in Rome). Its main seminary, Moreau Seminary, is located on the campus across St.
Joseph lake from the Main Building. Old College, the oldest building on campus and located near the shore
of St. Mary lake, houses undergraduate seminarians. Retired priests and brothers reside in Fatima House (a
former retreat center), Holy Cross House, as well as Columba Hall near the Grotto. The university through
the Moreau Seminary has ties to theologian Frederick Buechner. While not Catholic, Buechner has praised
writers from Notre Dame and Moreau Seminary created a Buechner Prize for Preaching.']}
```

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)
```

```
0.248
```

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, "RAG QA with Dense Retrieval")
```

```
Evaluating QA instances: 100%|██████████| 250/250 [05:27<00:00,  1.31s/it]RAG QA with Dense Retrieval Evalu
Exact Match: 9.60%
F1 Score: 19.01%
```

```
ROUGE2 F1: 7.41%
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: According to the 2010 Census, segregation in Detroit has decreased in absolute and in rela
Exact Match: 0, F1 Score: 0.09999999999999999
ROUGE-2 F1-score: 0.0
----------------------------------------
Question: When did Outcault's The Yellow Kid appear in newspapers?
Gold Answer: 1890s
Predicted Answer: 1894
Exact Match: 0, F1 Score: 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 which granted home rule
Exact Match: 0, F1 Score: 0.0689655172413793
ROUGE-2 F1-score: 0.0
----------------------------------------
Question: Treating the mitrailleuse like what rendered it far less effective
Gold Answer: artillery
Predicted Answer: French gunners were not trained to use the mitrailleuse effectively due to its complexity
Exact Match: 0, F1 Score: 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.

```
# Experiment with different k values for overlap retriever
print("="*60)
print("OVERLAP RETRIEVER - VARYING K")
print("="*60)

k_values = [1, 5, 10, 20]
overlap_results = {}

for k in k_values:
    print(f"\nEvaluating with k={k}...")
    rag_qa_pairs_k = add_rag_context(evaluation_benchmark['qas'], candidate_contexts, retrieve_overlap, top_
    retriever_acc = evaluate_retriever(rag_qa_pairs_k)
    eval_results = evaluate_qa(rag_qa, rag_qa_pairs_k)

    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]

    overlap_results[k] = {
        'retriever_accuracy': retriever_acc * 100,
        'exact_match': sum(exact_matches) / len(exact_matches) * 100,
        'f1_score': sum(f1_scores) / len(f1_scores) * 100,
        'rouge2_f1': sum(rouge2_f1) / len(rouge2_f1) * 100
    }

    print(f"k={k}: Retriever Acc={overlap_results[k]['retriever_accuracy']:.2f}%, EM={overlap_results[k]['ex
```

```
============================================================
OVERLAP RETRIEVER - VARYING K
============================================================

Evaluating with k=1...
Retrieving contexts: 100%|██████████| 250/250 [08:16<00:00,  1.99s/it]
Evaluating QA instances: 100%|██████████| 250/250 [02:58<00:00,  1.40it/s]
k=1: Retriever Acc=36.80%, EM=15.60%, F1=28.27%, ROUGE-2=12.36%

Evaluating with k=5...
Retrieving contexts: 100%|██████████| 250/250 [08:19<00:00,  2.00s/it]
Evaluating QA instances: 100%|██████████| 250/250 [06:46<00:00,  1.63s/it]
k=5: Retriever Acc=52.00%, EM=24.80%, F1=37.71%, ROUGE-2=18.69%

Evaluating with k=10...
Retrieving contexts: 100%|██████████| 250/250 [08:14<00:00,  1.98s/it]
Evaluating QA instances: 100%|██████████| 250/250 [12:57<00:00,  3.11s/it]
k=10: Retriever Acc=58.80%, EM=24.40%, F1=37.98%, ROUGE-2=20.35%

Evaluating with k=20...
Retrieving contexts: 100%|██████████| 250/250 [08:15<00:00,  1.98s/it]
Evaluating QA instances: 100%|██████████| 250/250 [30:18<00:00,  7.28s/it]k=20: Retriever Acc=64.80%, EM=11
```

```
# Experiment with different k values for dense retriever
print("\n" + "="*60)
print("DENSE RETRIEVER - VARYING K")
print("="*60)
```

```
dense_results = {}

for k in k_values:
    print(f"\nEvaluating with k={k}...")
    rag_qa_pairs_k = add_rag_context(evaluation_benchmark['qas'], candidate_contexts,
                                     retrieve_cosine, question_embeddings, context_embeddings, top_k=k)
    retriever_acc = evaluate_retriever(rag_qa_pairs_k)
    eval_results = evaluate_qa(rag_qa, rag_qa_pairs_k)

    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]

    dense_results[k] = {
        'retriever_accuracy': retriever_acc * 100,
        'exact_match': sum(exact_matches) / len(exact_matches) * 100,
        'f1_score': sum(f1_scores) / len(f1_scores) * 100,
        'rouge2_f1': sum(rouge2_f1) / len(rouge2_f1) * 100
    }

    print(f"k={k}: Retriever Acc={dense_results[k]['retriever_accuracy']:.2f}%, EM={dense_results[k]['exac
```

```
============================================================
DENSE RETRIEVER - VARYING K
============================================================

Evaluating with k=1...
Retrieving contexts: 100%|██████████| 250/250 [00:29<00:00,  8.38it/s]
Evaluating QA instances: 100%|██████████| 250/250 [02:53<00:00,  1.44it/s]
k=1: Retriever Acc=10.80%, EM=4.80%, F1=11.91%, ROUGE-2=3.65%

Evaluating with k=5...
Retrieving contexts: 100%|██████████| 250/250 [00:29<00:00,  8.42it/s]
Evaluating QA instances: 100%|██████████| 250/250 [05:26<00:00,  1.31s/it]
k=5: Retriever Acc=24.80%, EM=9.60%, F1=19.01%, ROUGE-2=7.41%

Evaluating with k=10...
Retrieving contexts: 100%|██████████| 250/250 [00:29<00:00,  8.50it/s]
Evaluating QA instances: 100%|██████████| 250/250 [08:22<00:00,  2.01s/it]
k=10: Retriever Acc=32.40%, EM=12.40%, F1=23.05%, ROUGE-2=9.01%

Evaluating with k=20...
Retrieving contexts: 100%|██████████| 250/250 [00:29<00:00,  8.47it/s]
Evaluating QA instances:   0%|          | 1/250 [00:04<17:03,  4.11s/it]This is a friendly reminder - the cu
Evaluating QA instances: 100%|██████████| 250/250 [16:11<00:00,  3.88s/it]k=20: Retriever Acc=38.80%, EM=11
```

```
# Display results in a table format
import pandas as pd

print("\n" + "="*60)
print("SUMMARY TABLE: OVERLAP RETRIEVER")
print("="*60)

overlap_df = pd.DataFrame(overlap_results).T
overlap_df.index.name = 'k'
overlap_df.columns = ['Retriever Acc (%)', 'Exact Match (%)', 'F1 Score (%)', 'ROUGE-2 F1 (%)']
print(overlap_df.to_string(float_format='%.2f'))

print("\n" + "="*60)
print("SUMMARY TABLE: DENSE RETRIEVER")
print("="*60)

dense_df = pd.DataFrame(dense_results).T
dense_df.index.name = 'k'
dense_df.columns = ['Retriever Acc (%)', 'Exact Match (%)', 'F1 Score (%)', 'ROUGE-2 F1 (%)']
print(dense_df.to_string(float_format='%.2f'))
```

```
============================================================
SUMMARY TABLE: OVERLAP RETRIEVER
============================================================
    Retriever Acc (%)  Exact Match (%)  F1 Score (%)  ROUGE-2 F1 (%)
k
1               36.80            15.60         28.27           12.36
5               52.00            24.80         37.71           18.69
10              58.80            24.40         37.98           20.35
20              64.80            11.20         18.69           10.10

============================================================
SUMMARY TABLE: DENSE RETRIEVER
============================================================
    Retriever Acc (%)  Exact Match (%)  F1 Score (%)  ROUGE-2 F1 (%)
k
1               10.80             4.80         11.91            3.65
5               24.80             9.60         19.01            7.41
10              32.40            12.40         23.05            9.01
20              38.80            11.60         22.47            9.55
```

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).

Novel Idea: Hybrid Retrieval with Score Fusion

For this improvement, I will implement a **hybrid retrieval approach** that combines both word overlap and dense (BERT-based) retrieval using score fusion. The key ideas are:

1. **Complementary Strengths**: Word overlap captures exact keyword matches (good for named entities, dates, etc.), while dense retrieval captures semantic similarity (good for paraphrases and concepts).
2. **Reciprocal Rank Fusion**: Instead of simple score averaging, I'll use Reciprocal Rank Fusion (RRF), which has been shown to be effective in information retrieval. RRF combines rankings from multiple systems without requiring score normalization.
3. **Re-ranking**: The hybrid approach retrieves more candidates (e.g., top-10 from each method) and re-ranks them based on combined scores.

The RRF score for a context is calculated as:

$$\text{RRF}(c) = \sum_{r \in R} \frac{1}{k + rank_r(c)}$$

where R is the set of retrieval methods, $rank_r(c)$ is the rank of context c in method r, and k is a constant (typically 60).

```
def retrieve_hybrid_rrf(question, question_emb, contexts, context_embeddings, top_k=5, k_constant=60):
    """
    Hybrid retrieval using Reciprocal Rank Fusion (RRF) to combine
    word overlap and dense retrieval rankings.
    """
    # Get rankings from both methods
    # Retrieve more candidates for re-ranking (e.g., top-20)
    candidate_k = min(20, len(contexts))

    # Word overlap retrieval
    question_tokens = set(get_tokens(question))
    overlap_scores = []
    for context in contexts:
        context_tokens = set(get_tokens(context))
        overlap = len(question_tokens & context_tokens)
        overlap_scores.append(overlap)

    overlap_ranking = sorted(range(len(overlap_scores)),
                            key=lambda i: overlap_scores[i],
                            reverse=True)

    # Dense retrieval
    if question_emb.dim() == 1:
        question_emb = question_emb.unsqueeze(0)

    similarities = F.cosine_similarity(question_emb, context_embeddings, dim=1)
    dense_ranking = torch.argsort(similarities, descending=True).cpu().numpy()

    # Compute RRF scores
    rrf_scores = {}

    # Add scores from overlap ranking
    for rank, idx in enumerate(overlap_ranking[:candidate_k]):
        if idx not in rrf_scores:
            rrf_scores[idx] = 0
        rrf_scores[idx] += 1.0 / (k_constant + rank)

    # Add scores from dense ranking
    for rank, idx in enumerate(dense_ranking[:candidate_k]):
        if idx not in rrf_scores:
            rrf_scores[idx] = 0
        rrf_scores[idx] += 1.0 / (k_constant + rank)

    # Sort by RRF score and get top-k
    top_k_indices = sorted(rrf_scores.keys(), key=lambda i: rrf_scores[i], reverse=True)[:top_k]

    return [contexts[i] for i in top_k_indices]
```

```
# Modified add_rag_context for hybrid retrieval
def add_rag_context_hybrid(qa_items, contexts, question_embeddings, context_embeddings, top_k=5):
    result_items = copy.deepcopy(qa_items)

    for i, inst in tqdm.tqdm(enumerate(result_items), desc="Retrieving contexts (hybrid)", total=len(resul
        question = inst['question']
        question_emb = question_embeddings[i]
        retrieved_contexts = retrieve_hybrid_rrf(question, question_emb, contexts,
                                                  context_embeddings, top_k)
        inst['rag_contexts'] = retrieved_contexts
    return result_items
```

```
# Evaluate hybrid retrieval with k=5
print("="*60)
print("HYBRID RETRIEVAL EVALUATION (k=5)")
print("="*60)

hybrid_qa_items = add_rag_context_hybrid(evaluation_benchmark['qas'], candidate_contexts,
                                         question_embeddings, context_embeddings, top_k=5)

# Check retriever accuracy
hybrid_retriever_acc = evaluate_retriever(hybrid_qa_items)
print(f"\nHybrid Retriever Accuracy: {hybrid_retriever_acc * 100:.2f}%")

# Evaluate end-to-end QA performance
hybrid_eval_results = evaluate_qa(rag_qa, hybrid_qa_items)
present_results(hybrid_eval_results, "Hybrid RAG QA")
```

```
============================================================
HYBRID RETRIEVAL EVALUATION (k=5)
============================================================
Retrieving contexts (hybrid): 100%|██████████| 250/250 [08:40<00:00,  2.08s/it]

Hybrid Retriever Accuracy: 55.20%
Evaluating QA instances: 100%|██████████| 250/250 [05:58<00:00,  1.44s/it]Hybrid RAG QA Evaluation Results:
Exact Match: 20.00%
F1 Score: 34.05%
ROUGE2 F1: 16.11%
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: 30.1% non-Hispanic Black or African American, 10.9% non-Hispanic Asian, 3.4% non-Hispanic
Exact Match: 0, F1 Score: 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 which granted home rul
Exact Match: 0, F1 Score: 0.0689655172413793
ROUGE-2 F1-score: 0.0
----------------------------------------
Question: Treating the mitrailleuse like what rendered it far less effective
Gold Answer: artillery
Predicted Answer: French gunners were treated like artillery and in this role it was ineffective.
Exact Match: 0, F1 Score: 0.14285714285714288
ROUGE-2 F1-score: 0.0
----------------------------------------
```

```
# Compare all three retrieval methods side-by-side
print("\n" + "="*60)
print("COMPARISON: OVERLAP vs DENSE vs HYBRID (k=5)")
print("="*60)

comparison_data = {
    'Overlap': {
        'Retriever Acc (%)': overlap_results[5]['retriever_accuracy'],
        'Exact Match (%)': overlap_results[5]['exact_match'],
        'F1 Score (%)': overlap_results[5]['f1_score'],
        'ROUGE-2 F1 (%)': overlap_results[5]['rouge2_f1']
    },
    'Dense': {
        'Retriever Acc (%)': dense_results[5]['retriever_accuracy'],
        'Exact Match (%)': dense_results[5]['exact_match'],
        'F1 Score (%)': dense_results[5]['f1_score'],
        'ROUGE-2 F1 (%)': dense_results[5]['rouge2_f1']
    },
    'Hybrid': {
        'Retriever Acc (%)': hybrid_retriever_acc * 100,
        'Exact Match (%)': sum([res['exact_match'] for res in hybrid_eval_results]) / len(hybrid_eval_resu
        'F1 Score (%)': sum([res['f1_score'] for res in hybrid_eval_results]) / len(hybrid_eval_results) *
        'ROUGE-2 F1 (%)': sum([res['rouge2_f1'] for res in hybrid_eval_results]) / len(hybrid_eval_results
    }
}

comparison_df = pd.DataFrame(comparison_data).T
print(comparison_df.to_string(float_format='%.2f'))
```

```
============================================================
COMPARISON: OVERLAP vs DENSE vs HYBRID (k=5)
============================================================
         Retriever Acc (%)  Exact Match (%)  F1 Score (%)  ROUGE-2 F1 (%)
```

```
    Overlap             52.00            24.80         37.71           18.69
    Dense               24.80             9.60         19.01            7.41
    Hybrid              55.20            20.00         34.05           16.11
```

Analysis

The hybrid retrieval approach using Reciprocal Rank Fusion (RRF) combines the strengths of both word overlap and dense retrieval:

Key Benefits:

1. **Complementary Information**: Word overlap captures exact matches (useful for named entities, specific terms), while dense retrieval captures semantic similarity (useful for paraphrased questions).
2. **Robust Ranking**: RRF is a proven fusion method that doesn't require score normalization and is less sensitive to outliers than simple averaging.
3. **Improved Retriever Accuracy**: By combining both methods, we should see improved retrieval accuracy, which translates to better QA performance.

Expected Results:

- The hybrid approach should achieve retriever accuracy between or better than both individual methods
- End-to-end QA metrics (EM, F1, ROUGE-2) should improve when the correct context is retrieved more reliably
- The method is particularly beneficial when one retrieval method fails but the other succeeds