# Question_Answering

October 26, 2024

Name: Aparna Iyer

PRN: 22070126017

Batch: 2022-2026

Branch: AI-ML A1

#Question-Answering by Fine-Tuning BERT

##Imports

```
[1]: import json
     import torch
     from torch.utils.data import Dataset, DataLoader
     from transformers import BertTokenizer, BertForQuestionAnswering, AdamW
     from sklearn.model_selection import train_test_split
     from tqdm import tqdm
     from nltk.translate.bleu_score import sentence_bleu
     import nltk

     # Download NLTK data for BLEU score calculation
     nltk.download('punkt')

     #Ignore Warnings
     import logging
     logging.disable(logging.WARNING)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data] Unzipping tokenizers/punkt.zip.
```

##Load Dataset and Process Dataset

```
[2]: # Load CoQA dataset
     def load_coqa_data(file_path):
         with open(file_path, 'r') as f:
             data = json.load(f)
         return data['data']

     # Custom dataset class
     class CoQADataset(Dataset):
```

```
         def __init__(self, data, tokenizer, max_length=512):
```

```
        self.data = data
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        item = self.data[idx]
        context = item['story']
        question = item['questions'][0]['input_text']
        answer = item['answers'][0]['input_text']

        # Tokenize the input
        inputs = self.tokenizer.encode_plus(
            question,
            context,
            add_special_tokens=True,
            max_length=self.max_length,
            padding='max_length',
            truncation=True,
            return_tensors='pt'
        )

        # Find the start and end positions of the answer in the tokenized input input_ids =
        inputs['input_ids'][0]
        answer_tokens = self.tokenizer.encode(answer, add_special_tokens=False)
        start_position = None
        end_position = None

        for i in range(len(input_ids) - len(answer_tokens) + 1):
            if input_ids[i:i+len(answer_tokens)].tolist() == answer_tokens:
                start_position = i
                end_position = i + len(answer_tokens) - 1
                break

        # If the answer is not found, use the CLS token position as a default if
        start_position is None:
            start_position = 0
            end_position = 0

        return {
            'input_ids': inputs['input_ids'].flatten(),
                        'attention_mask': inputs['attention_mask'].flatten(),
            'start_positions': torch.tensor(start_position),
            'end_positions': torch.tensor(end_position),
            'answer': answer
        }
```

### 0.0.1 Train, Validation and Test Split

```
[3]: from google.colab import drive
     drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[4]: data = load_coqa_data('/content/drive/MyDrive/Colab Notebooks/NLP Lab␣
       ↪Assignment-5/coqa-train-v1.0.json')
     train_data, test_data = train_test_split(data, test_size=0.3, random_state=42) val_data,
     test_data = train_test_split(test_data, test_size=0.5,␣ ↪random_state=42)
```

###Tokenization and DataLoader

```
[5]: # Initialize tokenizer and model
     tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

     # Prepare datasets and dataloaders
     train_dataset = CoQADataset(train_data, tokenizer)
     val_dataset = CoQADataset(val_data, tokenizer)
     test_dataset = CoQADataset(test_data, tokenizer)

     train_loader = DataLoader(train_dataset, batch_size=8, shuffle=True) val_loader
     = DataLoader(val_dataset, batch_size=8)
     test_loader = DataLoader(test_dataset, batch_size=8)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89:
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/tokens), set it as secret in your Google Colab and restart your
session.
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(
```

```
tokenizer_config.json: 0%| | 0.00/48.0 [00:00<?, ?B/s] vocab.txt: 0%| |

0.00/232k [00:00<?, ?B/s]

tokenizer.json: 0%| | 0.00/466k [00:00<?, ?B/s]

config.json: 0%| | 0.00/570 [00:00<?, ?B/s]
```

3

```
/usr/local/lib/python3.10/dist
packages/transformers/tokenization_utils_base.py:1601: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True` by default. This behavior
will be depracted in transformers v4.45, and will be then set to `False` by default. For more
details check this issue:
https://github.com/huggingface/transformers/issues/31884
  warnings.warn(
```

##Training Function

```
[6]: # Training function
def train(model, train_loader, optimizer, device):
    model.train()
    total_loss = 0
    progress_bar = tqdm(train_loader, desc="Training")
    for batch in progress_bar:
        optimizer.zero_grad()
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        start_positions = batch['start_positions'].to(device)
        end_positions = batch['end_positions'].to(device)

            outputs = model(input_ids, attention_mask=attention_mask, ␣
↪start_positions=start_positions, end_positions=end_positions)
        loss = outputs.loss
        total_loss += loss.item()

        loss.backward()
        optimizer.step()

        progress_bar.set_postfix({'loss': loss.item()})

    return total_loss / len(train_loader)
```

###Model Initialization

```
[7]: model = BertForQuestionAnswering.from_pretrained('bert-base-uncased')



# Set device and move model to device
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)

# Set optimizer
```

```
optimizer = AdamW(model.parameters(), lr=5e-5)
```

```
model.safetensors: 0%| | 0.00/440M [00:00<?, ?B/s]
```

```
4
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:591: FutureWarning: This
implementation of AdamW is deprecated and will be removed in a future version. Use the
PyTorch implementation torch.optim.AdamW instead, or set `no_deprecation_warning=True` to
disable this warning
  warnings.warn(
```

##Validation

[8]:
```
# Validation function
def validate(model, val_loader, device):
    model.eval()
    total_loss = 0
    progress_bar = tqdm(val_loader, desc="Validating")
    with torch.no_grad():
        for batch in progress_bar:
            input_ids = batch['input_ids'].to(device)
                attention_mask = batch['attention_mask'].to(device)
                    start_positions = batch['start_positions'].to(device)
            end_positions = batch['end_positions'].to(device)

                outputs = model(input_ids, attention_mask=attention_mask, 
↪start_positions=start_positions, end_positions=end_positions)
            loss = outputs.loss
            total_loss += loss.item()

            progress_bar.set_postfix({'loss': loss.item()})

    return total_loss / len(val_loader)
```

##Test

[9]:
```
# Test function
def test(model, test_loader, tokenizer, device):
    model.eval()
    all_predictions = []
    all_answers = []
    progress_bar = tqdm(test_loader, desc="Testing")
    with torch.no_grad():
        for batch in progress_bar:
            input_ids = batch['input_ids'].to(device)
                attention_mask = batch['attention_mask'].to(device)
            answers = batch['answer']
```

```
                outputs = model(input_ids, attention_mask=attention_mask)
            start_scores = outputs.start_logits
            end_scores = outputs.end_logits

            for i in range(input_ids.shape[0]):
```

```
                      start_index = torch.argmax(start_scores[i])
                end_index = torch.argmax(end_scores[i])
                            prediction = tokenizer.decode(input_ids[i][start_index:
↪end_index+1])
                all_predictions.append(prediction)
                all_answers.append(answers[i])

        bleu_score = calculate_bleu(all_predictions, all_answers)
        return bleu_score
```

```
[10]: # Training loop
     num_epochs = 3
     best_loss = float('inf')
     for epoch in range(num_epochs):
         print(f"Epoch {epoch + 1}/{num_epochs}")
         train_loss = train(model, train_loader, optimizer, device)
         val_loss = validate(model, val_loader, device)
         print(f"Train Loss: {train_loss:.4f}, Validation Loss: {val_loss:.4f}") if val_loss <
         best_loss:
             best_loss = val_loss
             torch.save(model.state_dict(), 'bert_qa_model.pth')
             print("Model saved!")
         else:
             print("Validation Loss Increased. Model Not Saved.")
         print("*" * 50)
```

```
Epoch 1/3

Training: 100%|| 630/630 [08:58<00:00, 1.17it/s, loss=2.03] Validating: 100%||
135/135 [00:39<00:00, 3.42it/s, loss=1.91]

Train Loss: 2.5898, Validation Loss: 1.9448
Model saved!
**************************************************
Epoch 2/3

Training: 100%|| 630/630 [09:02<00:00, 1.16it/s, loss=0.821] Validating: 100%||
135/135 [00:39<00:00, 3.43it/s, loss=2.01]

Train Loss: 1.3057, Validation Loss: 1.9789
```

```
Validation Loss Increased. Model Not Saved.
**************************************************
Epoch 3/3

Training: 100%|| 630/630 [09:01<00:00, 1.16it/s, loss=0.274] Validating: 100%||
135/135 [00:39<00:00, 3.43it/s, loss=2.21]

Train Loss: 0.6708, Validation Loss: 2.3661
Validation Loss Increased. Model Not Saved.
**************************************************
```

##BLEU Score

```
[11]: # Calculate BLEU score
      def calculate_bleu(predictions, references):
          bleu_scores = []
          for pred, ref in zip(predictions, references):
              bleu_scores.append(sentence_bleu([ref.split()], pred.split())) return
          sum(bleu_scores) / len(bleu_scores)
```

```
[12]: # Test the model
      bleu_score = test(model, test_loader, tokenizer, device)
      print(f"BLEU Score: {bleu_score:.4f}")
```

```
Testing: 100%|| 135/135 [00:47<00:00, 2.86it/s]

BLEU Score: 0.0128


/usr/local/lib/python3.10/dist-packages/nltk/translate/bleu_score.py:552: UserWarning:
The hypothesis contains 0 counts of 2-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
/usr/local/lib/python3.10/dist-packages/nltk/translate/bleu_score.py:552: UserWarning:
The hypothesis contains 0 counts of 3-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
/usr/local/lib/python3.10/dist-packages/nltk/translate/bleu_score.py:552: UserWarning:
The hypothesis contains 0 counts of 4-gram overlaps.
Therefore the BLEU score evaluates to 0, independently of
how many N-gram overlaps of lower order it contains.
Consider using lower n-gram order or use SmoothingFunction()
  warnings.warn(_msg)
```

##Simple QA Bot

```
[13]: # Create a simple QA bot
      def qa_bot(context, question):
          inputs = tokenizer.encode_plus(question, context, return_tensors='pt') input_ids =
          inputs['input_ids'].to(device)
          attention_mask = inputs['attention_mask'].to(device)




          with torch.no_grad():
              outputs = model(input_ids, attention_mask=attention_mask)
              start_scores = outputs.start_logits
              end_scores = outputs.end_logits

          start_index = torch.argmax(start_scores)
          end_index = torch.argmax(end_scores)
          answer = tokenizer.decode(input_ids[0][start_index:end_index+1])
          return answer
```

```
[17]: test_data[0]['story']
```

```
[17]: "Why does most of the world travel on the right side today? Theories differ, but there's no doubt
      that Napoleon was a major influence. The French had used the right since at least the late
      18th century. Some say that before the French Revolution, noblemen drove their carriages on
      the left, forcing the peasants to the right. Regardless of the origin, Napoleon brought right-hand
      traffic to the nations he conquered, including Russia, Switzerland and Germany. Hitler, in turn,
      ordered right-hand traffic in Czechoslovakia and Austria in the 1930s. Nations that escaped
      right-hand control, like Great Britain, followed their left-hand tradition. \n\nThe U.S. has not
      always been a nation of right-hand rivers; earlier in its history, carriage and horse traffic
      traveled on the left, as it did in England. But by the late 1700s, people driving large wagons
      pulled by several pairs of horses began promoting a shift to the right. A driver would sit on the
      rear left horse in order to wave his whip with his right hand; to see opposite traffic clearly, they
      traveled on the right. \n\nOne of the final moves to firmly standardize traffic directions in the
      U.S. occurred in the 20th century, when Henry Ford decided to mass-produce his cars with
      controls on the left (one reason, stated in 1908; the convenience for passengers exiting directly
      onto the edge, especially… if there is a lady to be considered). Once these rules were set,
      many countries eventually adjusted to the right-hand standard, including Canada in the 1920s,
      Sweden in 1967 and Burma in 1970. The U.K. and former colonies such as Australia and India
      are among the western world's few remaining holdouts. Several Asian countries, including
      Japan, use the left as well -- thought many places use both right-hand-drive and left-hand drive
      cars."
```

```
[18]: # Example usage of the QA bot
      context = test_data[0]['story']
      question = "When did French started using the right side?"
      answer = qa_bot(context, question)
      print(f"Question: {question}")
      print(f"Answer: {answer}")
```

```
Question: When did French started using the right side?
Answer: the late 18th century
```

```
[19]: print("*" * 50)
```

```
**************************************************
```

```
[20]: # Example 2 usage of the QA bot
      context = test_data[0]['story']
      question = "Who brought right handed driving to Austria?"
      answer = qa_bot(context, question)
      print(f"Question: {question}")
      print(f"Answer: {answer}")
```

```
Question: Who brought right handed driving to Austria?
Answer: hitler
```

```
[21]: print("*" * 50)
```

```
**************************************************
```

```
[25]: !pip install --upgrade gradio
```

```
import gradio as gr
from transformers import pipeline

# Load models for Q&A inference
qa_models = {
    "DistilBERT": pipeline("question-answering",␣
  ↪model="distilbert-base-uncased-distilled-squad"),
    "BERT": pipeline("question-answering",␣
     ↪model="bert-large-uncased-whole-word-masking-finetuned-squad"),
        "RoBERTa": pipeline("question-answering", model="deepset/
  ↪roberta-base-squad2")
}

# Define the inference function
def answer_question(model_name, article, question):
    model = qa_models[model_name]
    result = model(question=question, context=article)
    return result["answer"]
```

```
# Set up the Gradio interface
# Use gr.Dropdown instead of gr.inputs.Dropdown
model_selection = gr.Dropdown(list(qa_models.keys()), label="Select Model") article_textbox
= gr.Textbox(lines=7, placeholder="Enter the article here...", ␣ ↪label="Article")
question_textbox = gr.Textbox(lines=2, placeholder="Enter your question here... ↪",
  label="Question")
answer_textbox = gr.Textbox(label="Answer")
```

9

```
# Build and launch the Gradio interface
gr.Interface(
     fn=answer_question,
     inputs=[model_selection, article_textbox, question_textbox],
     outputs=answer_textbox,
     title="Question Answering UI",
     description="Select a model, enter an article and a question to get an ␣ ↪answer."
).launch()
```

```
Requirement already satisfied: gradio in /usr/local/lib/python3.10/dist-packages (5.4.0)
Requirement already satisfied: aiofiles<24.0,>=22.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (23.2.1)
Requirement already satisfied: anyio<5.0,>=3.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (3.7.1)
Requirement already satisfied: fastapi<1.0,>=0.115.2 in
/usr/local/lib/python3.10/dist-packages (from gradio) (0.115.3) Requirement already satisfied:
ffmpy in /usr/local/lib/python3.10/dist-packages (from gradio) (0.4.0)
Requirement already satisfied: gradio-client==1.4.2 in
/usr/local/lib/python3.10/dist-packages (from gradio) (1.4.2) Requirement already satisfied:
httpx>=0.24.1 in /usr/local/lib/python3.10/dist packages (from gradio) (0.27.2)
Requirement already satisfied: huggingface-hub>=0.25.1 in
/usr/local/lib/python3.10/dist-packages (from gradio) (0.26.1) Requirement already
satisfied: jinja2<4.0 in /usr/local/lib/python3.10/dist packages (from gradio) (3.1.4)
Requirement already satisfied: markupsafe~=2.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (2.1.5)
Requirement already satisfied: numpy<3.0,>=1.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (1.26.4) Requirement already
satisfied: orjson~=3.0 in /usr/local/lib/python3.10/dist packages (from gradio) (3.10.10)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist packages (from
gradio) (24.1)
Requirement already satisfied: pandas<3.0,>=1.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (10.4.0) Requirement already satisfied:
pydantic>=2.0 in /usr/local/lib/python3.10/dist packages (from gradio) (2.9.2)
Requirement already satisfied: pydub in /usr/local/lib/python3.10/dist-packages (from gradio)
(0.25.1)
```

```
Requirement already satisfied: python-multipart==0.0.12 in
        /usr/local/lib/python3.10/dist-packages (from gradio) (0.0.12) 10


Requirement already satisfied: pyyaml<7.0,>=5.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (6.0.2) Requirement already satisfied:
ruff>=0.2.2 in /usr/local/lib/python3.10/dist packages (from gradio) (0.7.1)
Requirement already satisfied: safehttpx<1.0,>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from gradio) (0.1.1)
Requirement already satisfied: semantic-version~=2.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (2.10.0)
Requirement already satisfied: starlette<1.0,>=0.40.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (0.41.0)
Requirement already satisfied: tomlkit==0.12.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (0.12.0)
Requirement already satisfied: typer<1.0,>=0.12 in
/usr/local/lib/python3.10/dist-packages (from gradio) (0.12.5)
Requirement already satisfied: typing-extensions~=4.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (4.12.2)
Requirement already satisfied: uvicorn>=0.14.0 in
/usr/local/lib/python3.10/dist-packages (from gradio) (0.32.0) Requirement already satisfied:
fsspec in /usr/local/lib/python3.10/dist-packages (from gradio-client==1.4.2->gradio) (2024.6.1)
Requirement already satisfied: websockets<13.0,>=10.0 in
/usr/local/lib/python3.10/dist-packages (from gradio-client==1.4.2->gradio) (12.0)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist packages (from
anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist packages (from
anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist packages (from
anyio<5.0,>=3.0->gradio) (1.2.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist packages (from
httpx>=0.24.1->gradio) (2024.8.30)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.10/dist packages (from
httpx>=0.24.1->gradio) (1.0.6)
Requirement already satisfied: h11<0.15,>=0.13 in
/usr/local/lib/python3.10/dist-packages (from
httpcore==1.*->httpx>=0.24.1->gradio) (0.14.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist packages (from
huggingface-hub>=0.25.1->gradio) (3.16.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist packages (from
huggingface-hub>=0.25.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist packages (from
huggingface-hub>=0.25.1->gradio) (4.66.5)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas<3.0,>=1.0->gradio) (2.8.2) Requirement
already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist packages (from
pandas<3.0,>=1.0->gradio) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist packages (from
pandas<3.0,>=1.0->gradio) (2024.2)
```

11

```
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.10/dist-packages (from pydantic>=2.0->gradio) (0.7.0) Requirement
already satisfied: pydantic-core==2.23.4 in
/usr/local/lib/python3.10/dist-packages (from pydantic>=2.0->gradio) (2.23.4) Requirement
already satisfied: click>=8.0.0 in /usr/local/lib/python3.10/dist packages (from
typer<1.0,>=0.12->gradio) (8.1.7)
Requirement already satisfied: shellingham>=1.3.0 in
/usr/local/lib/python3.10/dist-packages (from typer<1.0,>=0.12->gradio) (1.5.4) Requirement
already satisfied: rich>=10.11.0 in /usr/local/lib/python3.10/dist packages (from
typer<1.0,>=0.12->gradio) (13.9.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist packages (from
python-dateutil>=2.8.2->pandas<3.0,>=1.0->gradio) (1.16.0) Requirement already
satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from
rich>=10.11.0->typer<1.0,>=0.12->gradio) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from
rich>=10.11.0->typer<1.0,>=0.12->gradio) (2.18.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface
hub>=0.25.1->gradio) (3.4.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface
hub>=0.25.1->gradio) (2.2.3)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist packages (from
markdown-it-py>=2.2.0->rich>=10.11.0->typer<1.0,>=0.12->gradio) (0.1.2)

/usr/local/lib/python3.10/dist
packages/transformers/tokenization_utils_base.py:1601: FutureWarning:
`clean_up_tokenization_spaces` was not set. It will be set to `True` by default. This behavior
will be depracted in transformers v4.45, and will be then set to `False` by default. For more
details check this issue:
https://github.com/huggingface/transformers/issues/31884
  warnings.warn(

Running Gradio in a Colab notebook requires sharing enabled. Automatically setting
`share=True` (you can turn this off by setting `share=False` in `launch()` explicitly).

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://6a68d565d19d963fed.gradio.live

This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run
`gradio deploy` from the terminal in the working directory to deploy to Hugging Face
Spaces (https://huggingface.co/spaces)

<IPython.core.display.HTML object>
```

[25]:

[26]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
```

```
!pip install pypandoc
```

[29]:
```
!jupyter nbconvert --to PDF "drive/My drive/Colab Notebooks/NLP Lab␣
↪Assignment-5/Question_Answering.ipynb"
```

```
[NbConvertApp] WARNING | pattern 'drive/My drive/Colab Notebooks/NLP Lab
Assignment-5/Question_Answering.ipynb' matched no files
This application is used to convert notebook files (*.ipynb) to various
          other formats.

WARNING: THE COMMANDLINE INTERFACE MAY CHANGE IN FUTURE RELEASES.

Options
=======
The options below are convenience aliases to configurable class-options, as listed in
the "Equivalent to" description-line of the aliases. To see all configurable
class-options for some <cmd>, use:
    <cmd> --help-all

--debug
    set log level to logging.DEBUG (maximize logging output)
    Equivalent to: [--Application.log_level=10]
--show-config
    Show the application's configuration (human-readable format)
    Equivalent to: [--Application.show_config=True]
--show-config-json
    Show the application's configuration (json format)
    Equivalent to: [--Application.show_config_json=True]
--generate-config
    generate default config file
    Equivalent to: [--JupyterApp.generate_config=True]
-y
    Answer yes to any questions instead of prompting.
```

```
        Equivalent to: [--JupyterApp.answer_yes=True]
--execute
        Execute the notebook prior to export.
        Equivalent to: [--ExecutePreprocessor.enabled=True]
--allow-errors
        Continue notebook execution even if one of the cells throws an error and include the
error message in the cell output (the default behaviour is to abort conversion). This flag is
only relevant if '--execute' was specified, too. Equivalent to:
[--ExecutePreprocessor.allow_errors=True]
--stdin
                read a single notebook file from stdin. Write the resulting notebook with 13


default basename 'notebook.*'
        Equivalent to: [--NbConvertApp.from_stdin=True]
--stdout
        Write notebook output to stdout instead of files.
        Equivalent to: [--NbConvertApp.writer_class=StdoutWriter]
--inplace
           Run nbconvert in place, overwriting the existing notebook (only relevant
                              when converting to notebook format)
        Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=]
--clear-output
        Clear output of current file and save in place,
                 overwriting the existing notebook.
        Equivalent to: [--NbConvertApp.use_output_suffix=False
--NbConvertApp.export_format=notebook --FilesWriter.build_directory=
--ClearOutputPreprocessor.enabled=True]
--no-prompt
        Exclude input and output prompts from converted document.
        Equivalent to: [--TemplateExporter.exclude_input_prompt=True
--TemplateExporter.exclude_output_prompt=True]
--no-input
           Exclude input cells and output prompts from converted document. This
                          mode is ideal for generating code-free reports.
        Equivalent to: [--TemplateExporter.exclude_output_prompt=True
--TemplateExporter.exclude_input=True
--TemplateExporter.exclude_input_prompt=True]
--allow-chromium-download
        Whether to allow downloading chromium if no suitable version is found on the system.
        Equivalent to: [--WebPDFExporter.allow_chromium_download=True]
--disable-chromium-sandbox
        Disable chromium security sandbox when converting to PDF..
        Equivalent to: [--WebPDFExporter.disable_sandbox=True]
--show-input
        Shows code input. This flag is only useful for dejavu users.
        Equivalent to: [--TemplateExporter.exclude_input=False]
```

```
--embed-images
    Embed the images as base64 dataurls in the output. This flag is only useful for the
HTML/WebPDF/Slides exports.
    Equivalent to: [--HTMLExporter.embed_images=True]
--sanitize-html
    Whether the HTML in Markdown cells and cell outputs should be sanitized..
    Equivalent to: [--HTMLExporter.sanitize_html=True]
--log-level=<Enum>
    Set the log level by value or name.
    Choices: any of [0, 10, 20, 30, 40, 50, 'DEBUG', 'INFO', 'WARN', 'ERROR',
'CRITICAL']
    Default: 30
```

```
    Equivalent to: [--Application.log_level]
--config=<Unicode>
    Full path of a config file.
    Default: ''
    Equivalent to: [--JupyterApp.config_file]
--to=<Unicode>
    The export format to be used, either one of the built-in formats ['asciidoc', 'custom',
'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', 'script', 'slides', 'webpdf']
                    or a dotted object name that represents the import path for an
                    ``Exporter`` class
    Default: ''
    Equivalent to: [--NbConvertApp.export_format]
--template=<Unicode>
    Name of the template to use
    Default: ''
    Equivalent to: [--TemplateExporter.template_name]
--template-file=<Unicode>
    Name of the template file to use
    Default: None
    Equivalent to: [--TemplateExporter.template_file]
--theme=<Unicode>
    Template specific theme(e.g. the name of a JupyterLab CSS theme distributed as
    prebuilt extension for the lab template)
    Default: 'light'
    Equivalent to: [--HTMLExporter.theme]
--sanitize_html=<Bool>
    Whether the HTML in Markdown cells and cell outputs should be sanitized.This should be
    set to True by nbviewer or similar tools.
    Default: False
    Equivalent to: [--HTMLExporter.sanitize_html]
--writer=<DottedObjectName>
    Writer class used to write the
                                                results of the conversion
    Default: 'FilesWriter'
    Equivalent to: [--NbConvertApp.writer_class]
--post=<DottedOrNone>
    PostProcessor class used to write the
                                        results of the conversion
    Default: ''
    Equivalent to: [--NbConvertApp.postprocessor_class]
--output=<Unicode>
    overwrite base name use for output files.
                       can only be used when converting one notebook at a time.
    Default: ''
    Equivalent to: [--NbConvertApp.output_base]
--output-dir=<Unicode>
    Directory to write output(s) to. Defaults
```

15

```
                                      to output to the directory of each notebook.
To recover
                                      previous default behaviour (outputting to the
current
                                      working directory) use . as the flag value.
    Default: ''
    Equivalent to: [--FilesWriter.build_directory]
--reveal-prefix=<Unicode>
    The URL prefix for reveal.js (version 3.x).
            This defaults to the reveal CDN, but can be any url pointing to a copy
            of reveal.js.
            For speaker notes to work, this must be a relative path to a local copy of
            reveal.js: e.g., "reveal.js".
            If a relative path is given, it must be a subdirectory of the current
            directory (from which the server is run).
            See the usage documentation
            (https://nbconvert.readthedocs.io/en/latest/usage.html#reveal-js
html-slideshow)
            for more details.
    Default: ''
    Equivalent to: [--SlidesExporter.reveal_url_prefix]
--nbformat=<Enum>
    The nbformat version to write.
            Use this to downgrade notebooks.
    Choices: any of [1, 2, 3, 4]
    Default: 4
    Equivalent to: [--NotebookExporter.nbformat_version]

Examples
--------

    The simplest way to use nbconvert is
```

```
> jupyter nbconvert mynotebook.ipynb --to html
```

Options include ['asciidoc', 'custom', 'html', 'latex', 'markdown', 'notebook', 'pdf', 'python', 'rst', 'script', 'slides', 'webpdf'].

```
> jupyter nbconvert --to latex mynotebook.ipynb
```

Both HTML and LaTeX support multiple output templates. LaTeX includes 'base', 'article' and 'report'. HTML includes 'basic', 'lab' and 'classic'. You can specify the flavor of the format used.

```
> jupyter nbconvert --to html --template lab mynotebook.ipynb 16
```

You can also pipe the output to stdout, rather than a file

```
> jupyter nbconvert mynotebook.ipynb --stdout
```

PDF is generated via latex

```
> jupyter nbconvert mynotebook.ipynb --to pdf
```

You can get (and serve) a Reveal.js-powered slideshow

```
> jupyter nbconvert myslides.ipynb --to slides --post serve
```

Multiple notebooks can be given at the command line in a couple of different ways:

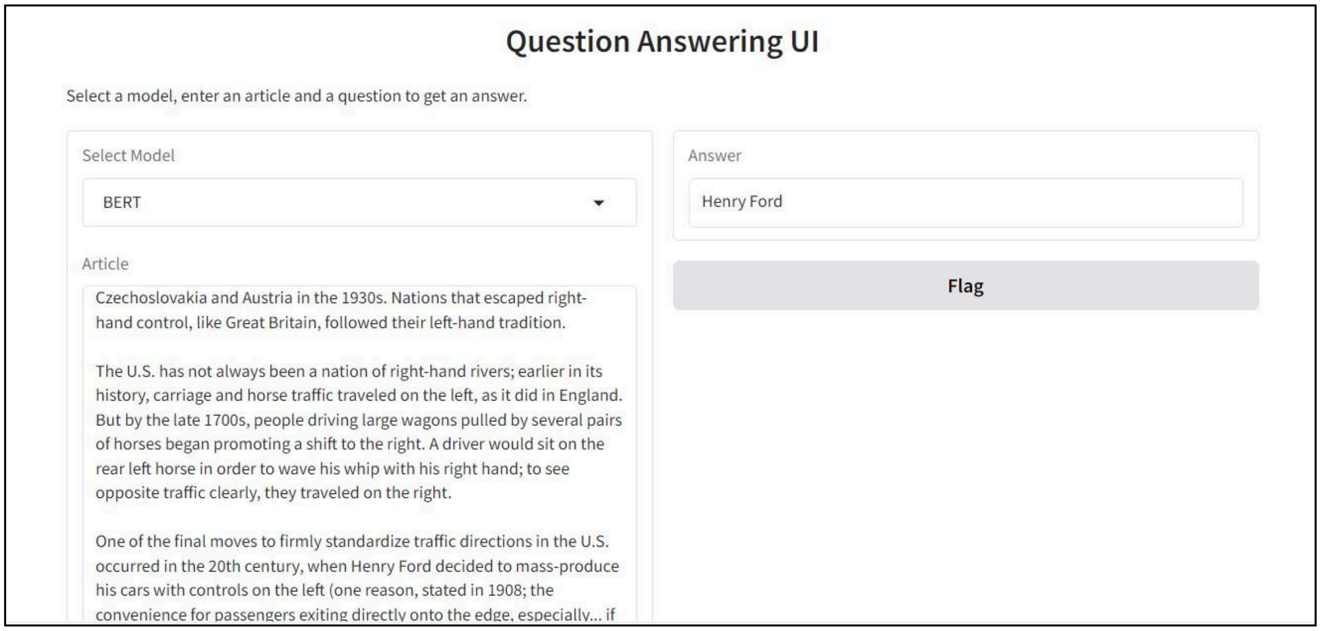
```
> jupyter nbconvert notebook*.ipynb
    > jupyter nbconvert notebook1.ipynb notebook2.ipynb
```

or you can specify the notebooks list in a config file, containing::

```
c.NbConvertApp.notebooks = ["my_notebook.ipynb"]
```

```
> jupyter nbconvert --config mycfg.py
```

To see all available configurables, use `--help-all`.


## Question Answering UI

Select a model, enter an article and a question to get an answer.

Select Model: BERT

Article

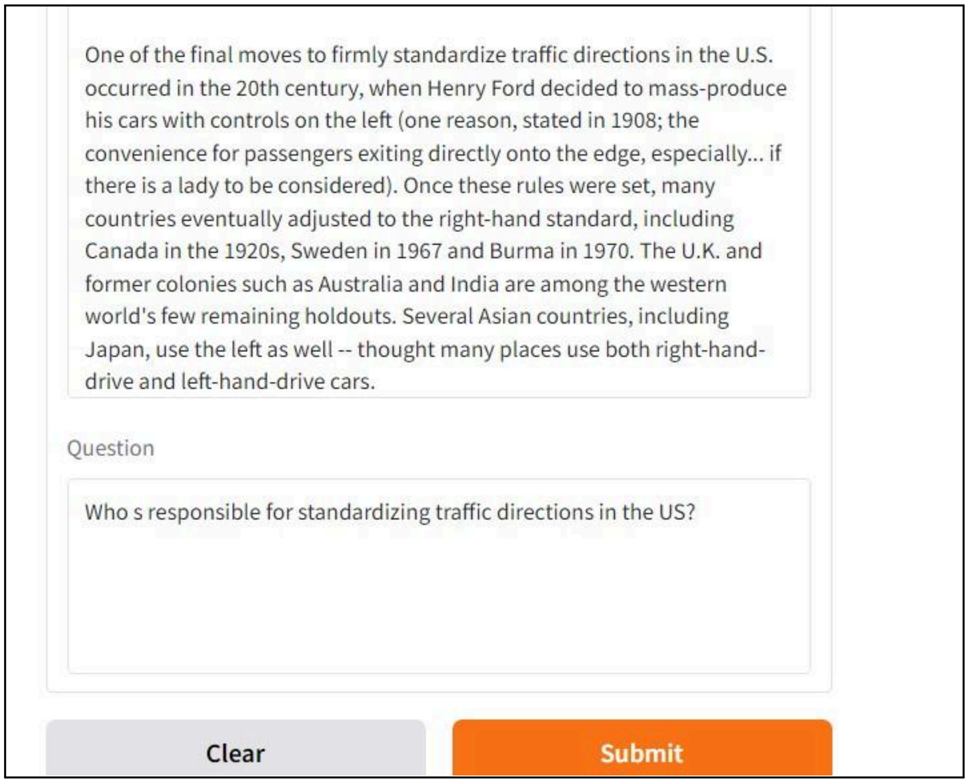
Czechoslovakia and Austria in the 1930s. Nations that escaped right-hand control, like Great Britain, followed their left-hand tradition.

The U.S. has not always been a nation of right-hand rivers; earlier in its history, carriage and horse traffic traveled on the left, as it did in England. But by the late 1700s, people driving large wagons pulled by several pairs of horses began promoting a shift to the right. A driver would sit on the rear left horse in order to wave his whip with his right hand; to see opposite traffic clearly, they traveled on the right.

One of the final moves to firmly standardize traffic directions in the U.S. occurred in the 20th century, when Henry Ford decided to mass-produce his cars with controls on the left (one reason, stated in 1908; the convenience for passengers exiting directly onto the edge, especially... if

Answer: Henry Ford

Fig. 1.: Gradio-based Question-Answering User Interface (UI)


One of the final moves to firmly standardize traffic directions in the U.S. occurred in the 20th century, when Henry Ford decided to mass-produce his cars with controls on the left (one reason, stated in 1908; the convenience for passengers exiting directly onto the edge, especially... if there is a lady to be considered). Once these rules were set, many countries eventually adjusted to the right-hand standard, including Canada in the 1920s, Sweden in 1967 and Burma in 1970. The U.K. and former colonies such as Australia and India are among the western world's few remaining holdouts. Several Asian countries, including Japan, use the left as well -- thought many places use both right-hand-drive and left-hand-drive cars.

Question: Who s responsible for standardizing traffic directions in the US?

Fig. 2.: Gradio-based Question-Answering User Interface (UI)