Report:

Grant Lewis

Saul Ramirez

For this lab, we explore the use of transformers in Question Answering systems to Extract data. For our project, we chose to deveolop a system that can answer trivia questions about the Pokemon pokedex.

Data used in this system was parsed from the PokeAPI, which hosts data for the Bulbapedia website. The data of interest was the pokedex description of each individual pokemon, its species, and its ID number. The way in which we have set this system up, we can handle multiple choice questions.

The trivia questions developed for this application are specialized on facts that are available in the pokedex entires. They don't consider the evolution chains, facts about people in the pokemon universe, or items.

The system works by parsing any pokemon mentioned in the question. Once all pokemon have been identified, we look up all entries for the pokemon mentioned in question and pass the entries one by one as context along with the question. We analyze all results and save the prediction with the highest probability as the final answer.

The transformer used is Roberta trained on squad2. Once we have a final answer, we compare our prediction to the accepted answer using a zero-shot transformer to compare if the results are the same. This didn't work as well as hoped. However, the intent was to simulate the real-life situation when people want credit for a similar answer and the judge says "close enough we'll count it".

We experimented with different ways of inputing the the context such as feeding the entire pokedex entry, vs feeding it line by line to identify if the answer was in the sentence. We found that the better method depended on the question, signaling high variance in the models.

Furthermore we noticed that the model was highly sensitive to the wording of the question, and small adjustments could change the answer. Using the output logits, we found that the correct answer was almost always in the top 3 outputs, however, for the purpose of trivia, this wasn't good enough.

We think that to improve this model, it would be good to build a generative model fine tuned on all of the Bulbapedia text, and pokemon entries to understand the context of the questions. Because the domain is so specific and atypical, things like being "an electric type" is an uncommon wording in context of the real world. Therefore providing the context into the jargon could produce better answers and allow the model to answer more complicated questions about the pokemon universe.

Overall this project was really fun to work on, and could be further developed to automate live trivia events.

```
# !pip install pyperclip
#·import·pyperclip
```

```
import pandas as pd
import re

# df = pd.read_csv("./contexts.csv", index_col="name")
df = pd.read_csv("./contexts_2.csv", index_col="name")
df.head()

#0.032  0.090 0.963 0.003
```

	id	all_cleaned
name		
Bulbasaur	1	Bulbasaur: id: 1 - nickname: Seed Pokémon - ty...
Ivysaur	2	Ivysaur: id: 2 - nickname: Seed Pokémon - type...
Venusaur	3	Venusaur: id: 3 - nickname: Seed Pokémon - typ...
Charmander	4	Charmander: id: 4 - nickname: Lizard Pokémon -...
Charmeleon	5	Charmeleon: id: 5 - nickname: Flame Pokémon - ...

```
context = df.loc['Pidgeot']['all_cleaned']
print(context)
# pyperclip.copy(context)
```

```
Pidgeot: id: 18 - nickname: Bird Pokémon - type: normal and flying - height: 15 - weigh
```

```
df_questions = pd.read_csv("./QandA_2.csv")
```

```
!pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting transformers
  Downloading transformers-4.24.0-py3-none-any.whl (5.5 MB)
     |████████████████████████████████| 5.5 MB 14.9 MB/s
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages
Collecting huggingface-hub<1.0,>=0.10.0
  Downloading huggingface_hub-0.10.1-py3-none-any.whl (163 kB)
     |████████████████████████████████| 163 kB 70.9 MB/s
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packa
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
  Downloading tokenizers-0.13.2-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.wh
     |████████████████████████████████| 7.6 MB 52.3 MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/di
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.10.1 tokenizers-0.13.2 transformers-4.24.0
```

```python
from transformers import pipeline
import torch
```

```python
from transformers import AutoModelForQuestionAnswering, AutoTokenizer, pipeline
model_name = "deepset/roberta-base-squad2"
# a) Get predictions
nlp = pipeline('question-answering', model=model_name, tokenizer=model_name, device=0)
```

```
Downloading: 100%          571/571 [00:00<00:00, 5.61kB/s]
Downloading: 100%          496M/496M [00:16<00:00, 30.8MB/s]
Downloading: 100%          79.0/79.0 [00:00<00:00, 1.08kB/s]
Downloading: 100%          899k/899k [00:00<00:00, 786kB/s]
Downloading: 100%          456k/456k [00:00<00:00, 624kB/s]
```

```python
SPLIT_CONTEXT = True
split_on = ";"
num_res_to_return = 3 if SPLIT_CONTEXT else 1

def answer_questions(questions, answers, num_res_to_return=1, should_split_context=False,
  should_split_context = should_split_context and (split_on is not None and split_on != 0)

  di = dict()
  for i, (question, answer) in enumerate(zip(questions, answers)):
    # print(re.sub(r"[^A-Za-z ]", " ", question))
    pokemon = [j for j in re.sub(r"[^A-Za-z ]", " ", question).split() if j in df.index]
    # print(pokemon)
    # temp_answer = []
    temp_answer = dict()
    # vals = []
    for p in pokemon:
      text = df.loc[p]['all_cleaned']

      title, descs = text.split(split_on, 1)
      descs = descs.split(split_on) if should_split_context and len(pokemon) == 1 else [de

      # parts = [] if not should_split_context else text.split(split_on)
      # parts.append(text)
      # for j, pt in enumerate(range(1,len(parts))):
        # context = f"{parts[0]}; {pt}" if j < len(parts) - 1 else pt

      for pt in descs:
        context = f"{title}; {pt}"
        # print(context)
        ans = nlp(question = question, context = context)

        guess = ans['answer']

        vote_cnt = 0 if guess not in temp_answer else temp_answer[guess]['vote_cnt']
        if guess not in temp_answer or ans['score'] > temp_answer[guess]['score']:
          temp_answer[guess] = ans
          temp_answer[guess]['vote_cnt'] = vote_cnt
        temp_answer[guess]['vote_cnt'] += 1

        # if len(temp_answer) == 0 or ans['answer'] not in vals:
          # vals.append(ans['answer'])
          # temp_answer.append(ans)
        # elif ans['score'] > temp_answer
    temp_answer = temp_answer.values()
    temp_answer = sorted(temp_answer, key = lambda x: x['score'], reverse=True) # max(temp
    temp_answer += [{'score':0, 'start':-1, 'end':-1, 'answer':'', 'vote_cnt':0} for _ in
    temp_answer = temp_answer[:num_res_to_return]
    # print(temp_answer)
    di[i] = (temp_answer, answer)
    # print(context)
  # print(di)
  return di

results = answer_questions(df_questions['Question'].tolist(), df_questions['Answer'].tolis
results_split = answer_questions(df_questions['Question'].tolist(), df_questions['Answer']

```

```
  /usr/local/lib/python3.7/dist-packages/transformers/pipelines/base.py:1046: UserWarning
    UserWarning,
for k, (vs, t) in results.items():
  print(k, '  answer:', t, '  guesses:', [f"{v['answer']} - {v['score']:.2f} ({v['vote_cnt
```

```
0   answer: four inches   guesses: ['four inches - 0.63 (1)']
1   answer: Big Jaw   guesses: ['Big Jaw Pokémon - 0.86 (1)']
2   answer: electric   guesses: ['forest dwelling - 0.78 (1)']
3   answer: eyes   guesses: ['eyes - 0.18 (1)']
4   answer: Psychic   guesses: ['psychic - 0.75 (1)']
5   answer: Mach 2 speed   guesses: ['Mach 2 speed - 0.63 (1)']
6   answer: Unstable genetic makeup   guesses: ['due to the environment in which it live
7   answer: Temperature drops 10 degrees   guesses: ['It may be trying to lay a curse or
8   answer: A Pendulum   guesses: ['a pendulum - 0.43 (1)']
9   answer: Pidgey   guesses: ['Pidgey - 0.91 (1)']
10   answer: Sunspots   guesses: ['sunspots - 0.00 (1)']
11   answer: 84 lbs.   guesses: ['380 - 0.99 (1)']
12   answer: 107   guesses: ['107 - 0.98 (1)']
13   answer: Lugia   guesses: ['Lugia - 0.44 (1)']
14   answer: South America   guesses: ['South America - 0.96 (1)']
15   answer: Volcano   guesses: ['the spout of a volcano - 0.17 (1)']
16   answer: Once a Year   guesses: ['Once a year - 0.84 (1)']
17   answer: Its Mother's   guesses: ['its dead mother - 0.35 (1)']
18   answer: 18   guesses: ['18 - 0.40 (1)']
19   answer: 10   guesses: ['10 - 0.84 (1)']
20   answer: Its Skin   guesses: ['hardened magma - 0.45 (1)']
21   answer: 7 feet   guesses: ['almost 7 feet - 0.34 (1)']
22   answer: Power Plants   guesses: ['near power plants - 0.36 (1)']
23   answer: 3000 degrees   guesses: ['3,000 degrees Fahrenheit - 0.02 (1)']
24   answer: Water   guesses: ['water - 0.75 (1)']
25   answer: 795 lbs.   guesses: ['795 - 0.99 (1)']
```

```
# for i, a in di.items():
#   di[i] = (a[0][0], a[1])
```

```
from transformers import pipeline
classifier = pipeline("zero-shot-classification",
                      model="facebook/bart-large-mnli", device=0)

```

```
Downloading: 100%     1.15k/1.15k [00:00<00:00, 39.8kB/s]
Downloading: 100%     1.63G/1.63G [00:27<00:00, 44.9MB/s]
Downloading: 100%     26.0/26.0 [00:00<00:00, 696B/s]
Downloading: 100%     899k/899k [00:00<00:00, 2.02MB/s]
```

```
desired = "same meaning"
candidate_labels = [desired, "different"]

MATCH_CUT_OFF = 0.5

points = 0

for i, a  in results.items():
  # print(a)
  guess = a[0]['answer'] if not isinstance(a[0], list) else a[0][0]['answer']
  actual = a[1]
  comparison = [guess, actual]
  pt = 0
  if comparison[0].lower() == comparison[1].lower():
    pt = 1
  else:
    words = re.sub(r'[^a-z0-9 ]', '', guess.lower()).split()
    actuals = re.sub(r'[^a-z0-9 ]', '', actual.lower()).split()
    match_cnt = 0
    for w in words:
      if w in actuals:
        match_cnt += 1
    print(match_cnt)
    if (match_cnt / min(len(words), len(actuals))) > 0.5:
      pt = 1


  input = " verses ".join(comparison)
  if pt == 0:
    out = classifier(input, candidate_labels)
    pt = 1 if out["labels"][0] == desired else 0
  points += pt
  print(f'{pt} / 1: {input}')

percentage = points / len(results)
print(f"Score: {percentage:.2%}")

```

```
1 / 1: four inches verses four inches
2
1 / 1: Big Jaw Pokémon verses Big Jaw
0
0 / 1: forest dwelling verses electric
1 / 1: eyes verses eyes
1 / 1: psychic verses Psychic
1 / 1: Mach 2 speed verses Mach 2 speed
0
0 / 1: due to the environment in which it lives verses Unstable genetic makeup
0
/usr/local/lib/python3.7/dist-packages/transformers/pipelines/base.py:1046: UserWarning
  UserWarning,
0 / 1: It may be trying to lay a curse on you verses Temperature drops 10 degrees
```

```
1 / 1: a pendulum verses A Pendulum
1 / 1: Pidgey verses Pidgey
1 / 1: sunspots verses Sunspots
0
0 / 1: 380 verses 84 lbs.
1 / 1: 107 verses 107
1 / 1: Lugia verses Lugia
1 / 1: South America verses South America
1
1 / 1: the spout of a volcano verses Volcano
1 / 1: Once a year verses Once a Year
1
0 / 1: its dead mother verses Its Mother's
1 / 1: 18 verses 18
1 / 1: 10 verses 10
0
0 / 1: hardened magma verses Its Skin
2
1 / 1: almost 7 feet verses 7 feet
2
1 / 1: near power plants verses Power Plants
2
1 / 1: 3,000 degrees Fahrenheit verses 3000 degrees
1 / 1: water verses Water
1
1 / 1: 795 verses 795 lbs.
Score: 76.92%
```

```
print(percentage)
```

```
0.9166666666666666
```