```
!pip install html2text --quiet
!pip install simpletransformers --quiet
```

```
!pip install -U ipykernel
!pip install modin[dask]
```

```
import numpy as np # Math
import requests # Getting text from websites
import html2text # Converting wiki pages to plain text
from googlesearch import search # To performing Google searches
import re
from simpletransformers.question_answering import QuestionAnsweringModel
from IPython.display import display
from IPython.html import widgets # Graphical display
from bs4 import BeautifulSoup
from markdown import markdown
```

```
/usr/local/lib/python3.6/dist-packages/IPython/html.py:14: ShimWarning: The `IPython.html` package has been deprecated
  "`IPython.html.widgets` has moved to `ipywidgets`.", ShimWarning)
```

```
# Avaliable models: https://huggingface.co/transformers/pretrained_models.html
model = QuestionAnsweringModel('distilbert', 'distilbert-base-uncased-distilled-squad')
```

```
Downloading: 100%                                   451/451 [00:00<00:00, 13.0kB/s]

Downloading: 100%                                   265M/265M [00:03<00:00, 75.9MB/s]

Downloading: 100%                                   232k/232k [00:00<00:00, 5.52MB/s]
```

```
question_data = {
  'qas':
  [{'question': 'What color is the sky?',
    'id': 0,
    'answers': [{'text': ' ', 'answer_start': 0}],
    'is_impossible': False}],
  'context': 'the sky is blue'
  }

prediction = model.predict([question_data])
print(prediction)
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 170.90it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 9198.04it/s]
Running Prediction: 100%                                        1/1 [00:00<00:00, 7.18it/s]
([{'id': 0, 'answer': ['blue', 'the sky is blue', 'sky is blue', 'is blue', 'the sky', 'the', 'sky', 'the sky is', '',
```

```
def predict_answer(model, question, contexts, seq_len=512, debug=False):
  split_context=[]
  if not isinstance(contexts, list):
    contexts=[]

  for context in contexts:
    for i in range(0, len(context), seq_len):
      split_context.append(context[i:i+seq_len])

  split_context= contexts

  f_data=[]

  for i,c in enumerate(split_context):
    f_data.append(
        {
        'qas':
    [{'question': question,
      'id': i,
      'answers':[{'text': ' ', 'answer_start':0}],
      'is_impossible':False}], # for unanswerable questions
    'context': c
        })
    prediction = model.predict(f_data)
    ans= prediction[0][0]['answer'][0]
    prob= prediction[1][0]['probability'][0]
    print("Answer: ",ans,", Probability: ",prob)

    # if debug:
    #   print(prediction)
    # preds= [x['answer'].lower().strip() for x in prediction if x['answer'].strip()!='']
    # if preds:
    #   return max(set(preds), key=preds.count)
    # return 'No answer'
```

```
predict_answer(model, 'what colour is sky?', ['the sky is blue in colour'])
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 178.51it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 9279.43it/s]
Running Prediction: 100%                                        1/1 [00:00<00:00, 20.38it/s]
Answer:  blue , Probability:  0.9413280059545852
```

```
predict_answer(model, 'which is the largest animal?', ['Although elephants are quite big but the blue whale is the largest an
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 156.49it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 9642.08it/s]
Running Prediction: 100%                                        1/1 [00:00<00:00, 19.28it/s]
Answer:  blue whale , Probability:  0.7292108232300563
```

```
predict_answer(model, 'would she go there?', ['Although she is really excited for it but due to the back pain she might not
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:00<00:00, 401.18it/s]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 8525.01it/s]
Running Prediction: 100%                                        1/1 [00:00<00:00, 22.98it/s]
Answer:  she might not go , Probability:  0.2112669022157816
```

```
links = list(search('what colour is the sky?', stop=2))

html_conv= html2text.HTML2Text()
```

```
html_conv.ignore_links= True
html_conv.escape_all= True

text=[]
for l in links:
  req= requests.get(l)
  text.append(html_conv.handle(req.text))
```

```
text
```

```
['�PNG \x1a \x00\x00\x00 IHDR\x00\x00\x00�\x00\x00\x00�\x08\x03\x00\x00\x00bH��\x00\x00\x01wPLTE���N����\n\x0
 '\n\n# Chicago Sky\n\nFrom Wikipedia, the free encyclopedia\n\nJump to navigation Jump to search\n\nChicago Sky  \n---
```

```
# Source: https://gist.github.com/lorey/eb15a7f3338f959a78cc3661fbc255fe
def markdown_to_text(markdown_string):
    """ Converts a markdown string to plaintext """

    # md -> html -> text since BeautifulSoup can extract text cleanly
    html = markdown(markdown_string)

    # remove code snippets
    html = re.sub(r'<pre>(.*?)</pre>', ' ', html)
    html = re.sub(r'<code>(.*?)</code >', ' ', html)

    # extract text
    soup = BeautifulSoup(html, "html.parser")
    text = ''.join(soup.findAll(text=True))

    return text

def format_text(text):
    text = markdown_to_text(text)
    text = text.replace('\n', ' ')
    return text
```

```
links = list(search('what color is the sky?', stop=2))
print(links)

html_conv = html2text.HTML2Text()
html_conv.ignore_links = True
html_conv.escape_all = True

text = []
for link in links:
    req = requests.get(link)
    text.append(html_conv.handle(req.text))
    text[-1] = format_text(text[-1])

print(text)
```

```
['https://www.universetoday.com/74020/what-color-is-the-sky/', 'https://science.discoveryplace.org/blog/ever-wonder-why
['503 Service Temporarily Unavailable  nginx', "Skip to main Content ADVANCED RESERVATION REQUIRED MAKE A RESERVATION A
```

```
def query_pages(query, n=5):
    return list(search(query, stop=n))

query_pages('Beyonce')
```

```
['http://www.beyonce.com/',
 'https://en.wikipedia.org/wiki/Beyonc%C3%A9',
 'https://www.beyonce.com/',
 'https://www.beyonce.com/article/beygood-housing-assistance/',
 'https://www.beyonce.com/tour/']
```

```
def query_to_text(query, n=5):
    html_conv = html2text.HTML2Text()
    html_conv.ignore_links = True
    html_conv.escape_all = True

    text = []
    for link in query_pages(query, n):
        req = requests.get(link)
        text.append(html_conv.handle(req.text))
        text.append(html_conv.handle(req.text))
        text[-1] = format_text(text[-1])

    return text
```

```
question = 'where was Beyonce born?'
context = query_to_text(question, n=3)
pred = predict_answer(model, question, context)
print(pred)
```

```
convert squad examples to features: 100%|██████████| 1/1 [00:03<00:00,  3.67s/it]
add example index and unique id: 100%|██████████| 1/1 [00:00<00:00, 3165.51it/s]
Running Prediction: 100%                                              17/17 [00:00<00:00, 51.93it/s]
Answer:  her mother , Probability:  0.5052509025240643
convert squad examples to features: 100%|██████████| 2/2 [00:03<00:00,  1.91s/it]
add example index and unique id: 100%|██████████| 2/2 [00:00<00:00, 10082.46it/s]
Running Prediction: 100%                                              19/19 [00:00<00:00, 48.09it/s]
Answer:  her mother , Probability:  0.5052509025240643
convert squad examples to features: 100%|██████████| 3/3 [00:04<00:00,  1.35s/it]
add example index and unique id: 100%|██████████| 3/3 [00:00<00:00, 9532.51it/s]
Running Prediction: 100%                                              20/20 [00:00<00:00, 51.53it/s]
Answer:  her mother , Probability:  0.5052509025240643
None
```