Pretrained Transformer Models from Hugging Face

AHugging Face is an AI company that develops and hosts tools used to develop machine learning based application. Amongst those tools is a transformers library accessible through Python that are pretrained for a variety of NLP tasks and fine-tuned to suit specific types of text data. These models are open-source and can be accessed throguh the `transformers` library in Python.

In this demonstration, I will run through several of these transformer models which have been pretrained for:

- Sentiment Analysis
- Abstraction-based summarization
- Question-Answering
- Table Question-Answering

But you can find a larger list of models, organized by task at the Hugging Face website. Let's begin by doing a pip install of `transformers` and then loading the remaining dependencies. Note that downloading these models can take a few minutes, depending on connection, but this only needs to occur once.

```
!pip install transformers
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import torch
from transformers import pipeline,TapexTokenizer, BartForConditionalGeneration

import nltk
import re
from nltk.tokenize import sent_tokenize
from nltk.corpus import stopwords
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.38.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.13.4)
Requirement already satisfied: huggingface-hub<1.0,>=0.19.3 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.20.3)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.25.2)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (24.0)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2023.12.25)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.19,>=0.14 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.15.2)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.66.2)
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.19.3->transform
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.19.3-
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2024.2.2)
```

Text Classification - Sentiment Analysis

```
# Using BERT for sentiment analysis
sentiment_model = pipeline('sentiment-analysis', model="nlptown/bert-base-multilingual-uncased-sentiment")
```

```
config.json: 100%                      953/953 [00:00<00:00, 28.1kB/s]
pytorch_model.bin: 100%                669M/669M [00:07<00:00, 119MB/s]
tokenizer_config.json: 100%            39.0/39.0 [00:00<00:00, 1.07kB/s]
vocab.txt: 100%                        872k/872k [00:00<00:00, 8.67MB/s]
special_tokens_map.json: 100%          112/112 [00:00<00:00, 2.72kB/s]
```

```
fin = pd.read_csv("/content/finance_sentiment.csv", nrows = 200)
fin.head()
```

	Sentence	Sentiment
0	The GeoSolutions technology will leverage Bene...	positive
1	$ESI on lows, down $1.50 to $2.50 BK a real po...	negative
2	For the last quarter of 2010 , Componenta 's n...	positive
3	According to the Finnish-Russian Chamber of Co...	neutral
4	The Swedish buyout firm has sold its remaining...	neutral

Next steps: View recommended plots

```
# don't pass the 'max_seq_length'
def truncate_sequence(sequence, max_length):
    if len(sequence) > max_length:
        sequence = sequence[:max_length] # only use the max sequence length (don't go over)
    return sequence

# Apply function to Sentence column & apply sentiment model
fin['Hug_Sentiment'] = fin['Sentence'].apply(lambda x: sentiment_model(truncate_sequence(x,512))[0]['label'])


# Print the sentiment for each article
fin.head()
```

	Sentence	Sentiment	Hug_Sentiment
0	The GeoSolutions technology will leverage Bene...	positive	5 stars
1	$ESI on lows, down $1.50 to $2.50 BK a real po...	negative	1 star
2	For the last quarter of 2010 , Componenta 's n...	positive	1 star
3	According to the Finnish-Russian Chamber of Co...	neutral	1 star
4	The Swedish buyout firm has sold its remaining...	neutral	1 star

Next steps: View recommended plots

```
fin[fin['Hug_Sentiment'] == '1 star']
```

	Sentence	Sentiment	Hug_Sentiment
1	$ESI on lows, down $1.50 to $2.50 BK a real po...	negative	1 star
2	For the last quarter of 2010 , Componenta 's n...	positive	1 star
3	According to the Finnish-Russian Chamber of Co...	neutral	1 star
4	The Swedish buyout firm has sold its remaining...	neutral	1 star
5	$SPY wouldn't be surprised to see a green close	positive	1 star
...	...	...	...
193	The government has instead proposed an exchang...	neutral	1 star
194	Stora is due to release its fourth-quarter and...	neutral	1 star
195	Full-year operating result for 2008 was 3.6 mi...	neutral	1 star
196	$NQ got hit hard lower this AM --> looks like ...	negative	1 star
197	In 2008 , AVC Systemhaus had net sales of EUR ...	neutral	1 star

107 rows × 3 columns

```
fin['Sentence'][12]
```

```
'The subdivision made sales revenues last year of EUR 480.7 million EUR 414.9 million i
n 2008 , and operating profits of EUR 44.5 million EUR 7.4 million .'
```

Text Summarization

```
# Load transcript from file
transcript_file = '/content/Analytics Training - Text Analysis with Carly Fox_2023-06-15-1.txt'
with open(transcript_file, 'r') as file:
    transcript_text = file.read()


def filter_speaker(text, target_speaker):
    filtered_lines = [] # create empty list of filtered lines
    lines = text.split('\n') # split lines where there are new lines
    i = 0
    while i < len(lines): # when lines are longer than 0
        speaker = lines[i].strip() # remove extra whitespaces
        if speaker.startswith(target_speaker): # if the line begins with the target speaker
            dialogue = lines[i+1].strip() # save as dialogue
            if dialogue:
                filtered_lines.append(dialogue) # add dialogue to filtered_lines
        i += 2  # Skip to the next speaker line
    return '\n'.join(filtered_lines) # rejoin filtered_lines


def preprocess_text(text):
    # Tokenize into sentences
    sentences = sent_tokenize(text)

    # Remove punctuation and convert to lowercase
    #cleaned_sentences = [re.sub(r'[^\w\s]', '', sentence.lower()) for sentence in sentences]

    # Remove stopwords and empty sentences
    #stop_words = set(stopwords.words('english'))
    #cleaned_sentences = [sentence for sentence in cleaned_sentences if sentence and sentence not in stop_words]

    return cleaned_sentences

target_speaker = 'Carly Fox'
filtered_text = filter_speaker(transcript_text, target_speaker)


def truncate_text(text, max_tokens = 850): # add inputs and comments
    tokens = text.split()
    if len(tokens) > max_tokens:
        tokens = tokens[:max_tokens]
    return ' '.join(tokens)

summarizer = pipeline("summarization", model="knkarthick/MEETING_SUMMARY")
```

```
config.json: 100%                                          1.59k/1.59k [00:00<00:00, 91.8kB/s]
model.safetensors: 100%                                    1.63G/1.63G [00:20<00:00, 89.9MB/s]
tokenizer_config.json: 100%                                337/337 [00:00<00:00, 7.26kB/s]
vocab.json: 100%                                           798k/798k [00:00<00:00, 25.3MB/s]
merges.txt: 100%                                           456k/456k [00:00<00:00, 5.97MB/s]
tokenizer.json: 100%                                       1.36M/1.36M [00:00<00:00, 18.7MB/s]
special_tokens_map.json: 100%                              239/239 [00:00<00:00, 5.32kB/s]
```

```
shortened_text = truncate_text(filtered_text, 800)
print(shortened_text)
```

```
And today, we're going to be talking about topic modeling, which is a pretty cool unsupervised method. So as always, we'll start off wit
```

```
# set length of summary and apply to shortened text
summary = summarizer(shortened_text, max_length = 100)

print(summary)
```

```
[{'summary_text': "Today, I'm going to talk about topic modeling. Topic modeling is an important part of natural language processing and
```

Question-Answering System

```
# Initialize the QA pipeline
qa_pipeline = pipeline('question-answering', model='distilbert-base-uncased-distilled-squad')
```

```
config.json: 100%                                   451/451 [00:00<00:00, 34.3kB/s]
model.safetensors: 100%                             265M/265M [00:01<00:00, 158MB/s]
tokenizer_config.json: 100%                         28.0/28.0 [00:00<00:00, 1.64kB/s]
vocab.txt: 100%                                     232k/232k [00:00<00:00, 4.67MB/s]
tokenizer.json: 100%                                466k/466k [00:00<00:00, 12.3MB/s]
```

```
# Define the context and the question, limit of 512 tokens (between context, question and output)
context = r"""
Thanksgiving Point is a 501(c)(3) non-profit indoor and outdoor farm, garden,
and museum complex in Lehi, Utah, United States. Its five main attractions
include Ashton Gardens, Butterfly Biosphere, Farm Country, Museum of Ancient
Life, and Museum of Natural Curiosity. It also operates multiple dining options,
event spaces, and gift shops. Each year, approximately 2.8 million guests visit
Thanksgiving Point.
"""

question = "How many people go to thanksgiving point every year?"

# Get the answer
answer = qa_pipeline({'context': context,
                      'question': question})
print(answer)
```

```
{'score': 0.7091150283813477, 'start': 371, 'end': 382, 'answer': '2.8 million'}
```

Table Question Answering

```
tokenizer = TapexTokenizer.from_pretrained("microsoft/tapex-base-finetuned-wikisql")
model = BartForConditionalGeneration.from_pretrained("microsoft/tapex-base-finetuned-wikisql")
```

```
tokenizer_config.json: 100%                         1.18k/1.18k [00:00<00:00, 53.0kB/s]
vocab.json: 100%                                    899k/899k [00:00<00:00, 12.4MB/s]
merges.txt: 100%                                    456k/456k [00:00<00:00, 22.9MB/s]
special_tokens_map.json: 100%                       772/772 [00:00<00:00, 45.1kB/s]
config.json: 100%                                   1.68k/1.68k [00:00<00:00, 89.1kB/s]
pytorch_model.bin: 100%                             558M/558M [00:07<00:00, 40.9MB/s]
generation_config.json: 100%                        236/236 [00:00<00:00, 14.9kB/s]
```

```
table = pd.read_csv('/content/voters (1).csv')

table = table.applymap(lambda x: str(x) if not isinstance(x, str) else x) # convert all values to strings
table.head()
```

	ResponseID	Age	IncomeCat	MStatus	Religion	Homeowner	Defense	Healthcare	Privac
0	1	61	2	married	Agnostic	y	3	3	
1	2	26	1	married	Christian	n	2	5	
2	3	28	2	divorced	Jewish	n	2	3	
3	4	23	1	married	Christian	n	3	1	
4	5	25	2	married	Christian	y	5	3	

Next steps: View recommended plots

```
query = "Which candidate has greater VIntent?"
encoding = tokenizer(table=table, query=query, return_tensors="pt", max_length=1024, truncation=True)


outputs = model.generate(**encoding)

print(tokenizer.batch_decode(outputs, skip_special_tokens=True))
```

```
[' kang']
```

```
table['VIntent'].value_counts()
```

```
VIntent
Kang     151
Kodos    147
Name: count, dtype: int64
```