Large Language Models (LLMs): Democratise AI with Low-Code/No-Code for Text, Images, and Audio

Content generation and content analysis in 3 lines of python code

A hands-on 101 tutorial by students at the University of Southampton

James Pickford (Year 3 student)
Yifan Chang (Year 2 student)
Aswathy Ajaykumar (Year 2 student)

08 May 2024
**Number of Parameters**: The smaller models like BERT have millions of parameters. BERT with 340 million parameters and BLOOM with 176 billion parameters.
Pipelines to use LLMs from Google, Microsoft, Facebook, OpenAI, …

What are the pipelines?
What are the pipelines?

The pipelines are wrappers, i.e. python functions or classes.

There are task-specific pipelines for audio, computer vision, natural language processing (NLP), and multimodal tasks (e.g. text-to-video).
Large Language Models (LLMs)

Pipelines for Natural Language Processing (NLP)

text → tokenizer → model → post processing → predictions
Hugging Face pipelines

Using LLMs for various tasks

the **3 lines** of python code

- instantiate the pipeline

**What is the task? What is the model?**

- the **input data** for the pipeline

  a data sample or data instance

- the **output of the pipeline** with the data given

  the result of executing the pipeline
Text Classification: Sentiment Analysis (SA)
What is the task? What is the model?

```python
pipe = pipeline(task="sentiment-analysis", model='siebert/sentiment-roberta-large-english')
```

the input data for the pipeline

```python
# a vector (one-dimensional array) with 2 string values
text_data = ['I love you', 'I hate you']
```
Text Classification with LLMs

Hugging Face pipeline

the **3 lines** of python code

```python
# ------------
### Text Classification: Sentiment Analysis (SA)
# ------------

# --- --- --- install and load libraries
!pip install transformers
from transformers import pipeline

# --- --- --- Transformers pipeline: 3 lines of code
pipe = pipeline(task="sentiment-analysis", model='siebert/sentiment-roberta-large-english')
text_data = ['I love you', 'I hate you']  # a vector (one-dimensional array) with 2 string values
print(pipe(text_data))
# ------------
```

the **output of the pipeline** with the data given

the **result** of executing the pipeline
Text Classification with LLMs

Hugging Face pipeline

the 3 lines of python code

```python
# Text Classification: Sentiment Analysis (SA)

#--- --- --- install and load libraries
#pip install transformers
#from transformers import pipeline

#--- --- --- Transformers pipeline: 3 lines of code
pipe = pipeline(task="sentiment-analysis", model='siebert/sentiment-roberta-large-english')
text_data = ['I love you', 'I hate you'] # a vector (one-dimensional array) with 2 string values
print(pipe(text_data))
```

the output of the pipeline with the data given

```json
[{"label": 'POSITIVE', 'score': 0.998561680316925},
{"label": 'NEGATIVE', 'score': 0.9991401433944702}]
```
Token Classification: Named-Entity Recognition (NER)
the Hugging Face pipeline

the 3 lines of python code

instantiate the pipeline

What is the task? What is the model?

```python
model_path = "d4data/biomedical-ner-all"
pipe = pipeline(task="ner",
               model=AutoModelForTokenClassification.from_pretrained(model_path),
               tokenizer=AutoTokenizer.from_pretrained(model_path))

the input data for the pipeline

a data sample or data instance

text_data = "I have a mild headache"
```
Token Classification with LLMs

Hugging Face pipeline

the 3 lines of python code

```python
# --------------
## Token Classification: Named-entity recognition (NER)
# --------------

#--- --- --- install and load libraries
!pip install transformers
from transformers import pipeline, AutoTokenizer, AutoModelForTokenClassification

#--- --- --- preliminaries
model_path = "d4data/biomedical-ner-all"

#--- --- --- Transformers pipeline: 3 lines of code
pipe = pipeline(task="ner", model=AutoModelForTokenClassification.from_pretrained(model_path),
               tokenizer=AutoTokenizer.from_pretrained(model_path))

text_data = "I have a mild headache"

print(pipe(text_data))
```

the output of the pipeline with the data given

the result of executing the pipeline
Hugging Face pipeline

the **3 lines** of python code

```python
# Token Classification: Named-entity recognition (NER)

# --- --- --- install and load libraries
!pip install transformers
from transformers import pipeline, AutoTokenizer, AutoModelForTokenClassification

# --- --- --- preliminaries
model_path = "d4data/biomedical-ner-all"

# --- --- --- Transformers pipeline: 3 lines of code
pipe = pipeline(task="ner", model=AutoModelForTokenClassification.from_pretrained(model_path),
                tokenizer=AutoTokenizer.from_pretrained(model_path))

text_data = "I have a mild headache"
print(pipe(text_data))
```

the **output** of the pipeline with the data given

```json
[{'entity': 'B-Severity', 'score': 0.9997888, 'index': 4, 'word': 'mild', 'start': 9, 'end': 13},
 {'entity': 'B-Sign_symptom', 'score': 0.99993956, 'index': 5, 'word': 'headache', 'start': 14, 'end': 22}]
```
Text Classification: SA
Token Classification: NER

No-Code AI using the Hugging Face pipelines
(changing the prompt Not the code)
Text Generation with LLMs

Hugging Face pipeline

the 3 lines of python code

instantiate the pipeline

What is the task? What is the model?

```python
pipe = pipeline(task="text-generation",
               model='mistralai/Mistral-7B-Instruct-v0.2')
```

the input data for the pipeline

a data sample or data instance

```python
text_prompt = "Classify the text into neutral, negative or positive. Text: I hate you. Sentiment:"

text_prompt = "Return a list with the medical entities in the text. Text: I have a mild headache. Named entities:""
```
Zero shot (open vocabulary) image classification
Image Classification with LLMs

Hugging Face pipeline

The 3 lines of python code

Instantiate the pipeline

What is the task? What is the model?

```
pipe = pipeline(task="zero-shot-image-classification",
model="openai/clip-vit-large-patch14")
```

The input data for the pipeline

A data sample or data instance

```
```
Hugging Face pipeline

the 3 lines of python code

```python
# zero-shot-image-classification

#--- --- --- install and load libraries
!pip install transformers
!pip install pillow

import requests
from transformers import pipeline
from PIL import Image

#--- --- --- Transformers pipeline: 3 lines of code
pipe = pipeline(task="zero-shot-image-classification", model="openai/clip-vit-large-patch14")


print(pipe(img_data, candidate_labels=['building', 'column', 'car', 'animal', 'cat', 'flowers']))
```

the output of the pipeline with the data given

the result of executing the pipeline
Hugging Face pipeline

the 3 lines of python code

```python
### zero-shot-image-classification
#
#
# --- --- --- install and load libraries
# pip install transformers
# pip install pillow

import requests
from transformers import pipeline
from PIL import Image

# --- --- --- Transformers pipeline: 3 lines of code

pipe = pipeline(task="zero-shot-image-classification", model="openai/clip-vit-large-patch14")


print(pipe(img_data, candidate_labels=["building", "column", "car", "animal", "cat", "flowers"]))
```

the output of the pipeline with the data given

```json
[{'score': 0.7278850078582764, 'label': 'building'}, {'score': 0.25961214303970337, 'label': 'column'}, {'score': 0.004940211772918701, 'label': 'car'}, {'score': 0.0047919861972332, 'label': 'animal'}, {'score': 0.0025786142796278, 'label': 'cat'}, {'score': 0.00019203458214178681, 'label': 'flowers'}]
```
State-of-the-art of prompting

Can a **textual prompt** generate a **video**?

Can this be done with Hugging Face pipelines in **3 lines of code**?
Can a textual prompt generate a video? Yes

Can this be done with Hugging Face pipelines in 3 lines of code? Yes
text to video with multimodal LLMs
What is the task? What is the model?

the 3 lines of python code

instantiate the pipeline

pipe = DiffusionPipeline.from_pretrained("damo-vilab/text-to-video-ms-1.7b", torch_dtype=torch.float16, variant="fp16").to("cuda")

the input data for the pipeline

a data sample or data instance

text_prompt = "Darth Vader surfing a wave"
Task-specific pipeline

the 3 lines of python code

```python
### text to video
#--- --- --- install and load libraries
!pip install transformers accelerate diffusers

import torch
from diffusers import DiffusionPipeline
from diffusers.utils import export_to_video

#--- --- --- preliminaries
#--- access to Google drive
from google.colab import drive
drive.mount('/content/drive')

#--- path to save the video generated
v_path = '/content/drive/MyDrive/newTut101_files/DarthVader_ex.mp4'

#--- --- --- Transformers pipeline: 3 lines of code
pipe = DiffusionPipeline.from_pretrained("damo-vilab/text-to-video-ms-1.7b", torch_dtype=torch.float16, variant="fp16").to("cuda")
text_prompt = "Darth Vader surfing a wave"
video_frames = pipe(text_prompt, num_frames=32).frames[0]

#--- --- --- extra line: save the video
video_path = export_to_video(video_frames, fps=10, output_video_path=v_path)
```

example

Google Colab

Connected to
Python 3 Google Compute Engine
backend (GPU)
RAM: 3.16 GB/62.80 GB
Disk: 32.25 GB/201.23 GB

the output of the pipeline with the data given

the result of executing the pipeline
Task-specific pipeline

the 3 lines of python code

diricet input data
for the pipeline

text_prompt = "Darth Vader surfing a wave"

the output of the pipeline with the data given
Audio classification: Language Identification
Audio Classification with LLMs

Hugging Face pipeline

The 3 lines of python code

Instantiate the pipeline

What is the task? What is the model?

pipe = pipeline(task="audio-classification",
               model="facebook/mms-lid-4017")

The input data for the pipeline

A data sample or data instance

audio_data = audio_sample["audio"]["array"]
Audio Classification with LLMs

Reusing existing datasets

Example

I would like to pay my electricity bill using my card, can you please assist?

```python
# Example: Using existing datasets
# loading and playing audio files

# --- --- --- install and load libraries
!pip install datasets
from datasets import load_dataset, Audio

# --- --- --- load audio file
# first file from train dataset of Minds-14: e-banking speech dataset Intent Classification
minds14 = load_dataset("PolyAI/minds14", name="en-AU", split="train[:1]")
audio_files = minds14.cast_column("audio", Audio(sampling_rate=16_000))
audio_sample = audio_files[0]

# --- --- --- play audio file
from IPython.display import Audio
Audio(audio_sample["audio"]['array'], rate=audio_sample["audio"]['sampling_rate'])
```
Audio Classification with LLMs

Hugging Face pipeline

the 3 lines of python code

```python
# Audio Classification: Language Identification (LID)

# install and load libraries
!pip install transformers datasets
from transformers import pipeline
from datasets import load_dataset, Audio

# preliminaries
# first file from train dataset of Minds-14: e-banking speech dataset Intent Classification
minds14 = load_dataset("PolyAI/minds14", name="en-AU", split="train[:1]"
audio_files = minds14.cast_column("audio", Audio(sampling_rate=16_000))
audio_sample = audio_files[0]

# Transformers pipeline: 3 lines of code
pipe = pipeline(task="audio-classification", model="facebook/mms-lid-4017")
audio_data = audio_sample["audio"]["array"]
print(pipe(audio_data))
```

the output of the pipeline with the data given

the result of executing the pipeline
Audio Classification with LLMs

Hugging Face pipeline

the 3 lines of python code

```python
# Audio Classification: Language Identification (LID)

#--- --- --- install and load libraries
!pip install transformers datasets
from transformers import pipeline
from datasets import load_dataset, Audio

#--- --- --- preliminaries
# first file from train dataset of Minds-14: e-banking speech dataset Intent Classification
minds14 = load_dataset("PolyAI/minds14", name="en-AU", split="train[:1]")
audio_files = minds14.cast_column("audio", Audio(sampling_rate=16_000))
audio_samples = audio_files[0]

#--- --- --- Transformers pipeline: 3 lines of code
pipe = pipeline(task="audio-classification", model="facebook/mms-lid-4017")
audio_data = audio_samples['audio'][:array]
print(pipe(audio_data))
```

the output of the pipeline with the data given

```json
[{'score': 0.915595302581787, 'label': 'eng'}, {'score': 0.030366547405719757, 'label': 'mri'},
{'score': 0.0034650196321308613, 'label': 'cym'}, {'score': 0.0018054329557344317, 'label': 'haw'},
{'score': 0.0013475015293806791, 'label': 'lat'}]
```
Audio classification: Keyword Spotting
What is the task? What is the model?

The 3 lines of python code

```python
pipe = pipeline(task="audio-classification",
model="anton-l/xtreme_s_xlsr_300m_minds14")
```

The input data for the pipeline

```python
audio_data = audio_sample['audio']['array']
```
Reusing existing datasets

```python
# Using existing datasets:
## loading and playing audio files

### install and load libraries
!pip install datasets
from datasets import load_dataset, Audio

### load audio file
#### first file from train dataset of Minds-14: e-banking speech dataset Intent Classification
minds14 = load_dataset("PolyAI/minds14", name="en-AU", split="train[:1]")
audio_files = minds14.cast_column("audio", Audio(sampling_rate=16_000))
audio_sample = audio_files[0]

### play audio file
from IPython.display import Audio
Audio(audio_sample["audio"]["array", rate=audio_sample["audio"]["sampling_rate"])
```

I would like to pay my electricity bill using my card, can you please assist?
Hugging Face pipeline

the 3 lines of python code

```python
# Audio Classification: Keyword Spotting (KWS)
#
#--- --- --- install and load libraries
!pip install transformers datasets
from transformers import pipeline
from datasets import load_dataset, Audio

#--- --- --- preliminaries
### first file from train dataset of Minds-14: e-banking speech dataset Intent Classification
minds14 = load_dataset("PolyAI/minds14", name="en-AU", split="train[:1]")
audio_files = minds14.cast_column("audio", Audio(sampling_rate=16_000))
audio_sample = audio_files[0]

#--- --- --- Transformers pipeline: 3 lines of code
pipe = pipeline(task="audio-classification", model="anton-l/xtreme_s_xlsr_300m_minds14")
audio_data = audio_sample["audio"]['array']
print(pipe(audio_data))
```

the output of the pipeline with the data given

the result of executing the pipeline
Hugging Face pipeline

the 3 lines of python code

```python
# Audio Classification: Keyword Spotting (KWS)

#--- --- --- install and load libraries
!pip install transformers datasets
from transformers import pipeline
from datasets import load_dataset, Audio

#--- --- --- preliminaries
### first file from train dataset of Minds-14: e-banking speech dataset Intent Classification
minds14 = load_dataset("PolyAI/minds14", name="en-AU", split="train[:1]"
audio_files = minds14.cast_column("audio", Audio(sampling_rate=16_000))
audio_sample = audio_files[0]

#--- --- --- Transformers pipeline: 3 lines of code
pipe = pipeline(task="audio-classification", model="anton-1/xtreme_s_xlslr_300m_minds14")
audio_data = audio_sample["audio"]
print(pipe(audio_data))
```

the output of the pipeline with the data given

```json
[{'score': 0.962530791759491, 'label': 'pay_bill'}, {'score': 0.02867300808429718, 'label': 'freeze'}, {'score': 0.003349815495312214, 'label': 'card_issues'}, {'score': 0.002005813643336296, 'label': 'abroad'}, {'score': 0.0008484353311359882, 'label': 'high_value_payment'}]
```
What’s next?

Parameters (number of variables): from M (millions), to B (billions), to T (trillions)
Questions

Special thanks to:
Prof Max Bramer (Chair of BCS SGAI) and
Dhruv Khanna (President of AI Society at
University of Southampton)

M Arguello-Casteleiro (University of Southampton) and
T Furmston (University of Manchester) for the
Hands-on 101 Tutorial
Democratise AI with Low-Code/No-Code
for Text, Images, and Audio