GenAI-Semester-6

HandsOn Unit-1 **Date:** 23/01/2026

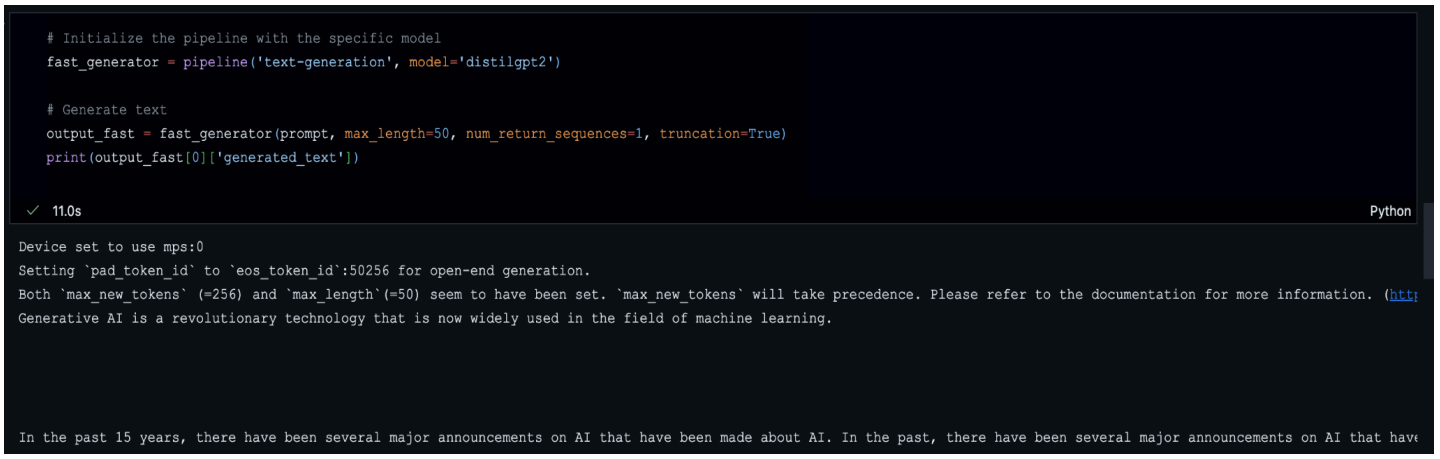
Name: Abhranshu Sarkar	**SRN:** PES2UG23CS024	**Section:** A

Transformers Library and `pipeline()`

The transformers library acts as the bridge between the HuggingFace platform and our code, as it provides APIs using which we can easily download, load and run many state-of-the-art models.
The `pipeline()` function helps abstract the complex steps involved with building and running a model into 3 steps, namely preprocessing, model inference and post-processing.

Dumb vs. Smart Models

The smaller model `distilgpt2` runs faster and uses less memory but it tends to sway more from the prompt/context as can be seen in the output below. It is our dumb model.



```
# Initialize the pipeline with the specific model
fast_generator = pipeline('text-generation', model='distilgpt2')

# Generate text
output_fast = fast_generator(prompt, max_length=50, num_return_sequences=1, truncation=True)
print(output_fast[0]['generated_text'])
```

✓ 11.0s Python

```
Device set to use mps:0
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Both `max_new_tokens` (=256) and `max_length`(=50) seem to have been set. `max_new_tokens` will take precedence. Please refer to the documentation for more information. (http
Generative AI is a revolutionary technology that is now widely used in the field of machine learning.



In the past 15 years, there have been several major announcements on AI that have been made about AI. In the past, there have been several major announcements on AI that have
```

The larger model `gpt2` uses more memory and runs slower but it gives much better and more coherent output as compared to the smaller model. It is our smart model.

```
smart_generator = pipeline('text-generation', model='gpt2')

output_smart = smart_generator(prompt, max_length=50, num_return_sequences=1, truncation=True)
print(output_smart[0]['generated_text'])
```

✓ 3.3s Python

```
Device set to use mps:0
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Both `max_new_tokens` (=256) and `max_length`(=50) seem to have been set. `max_new_tokens` will take precedence. Please refer to the documentation for more information. (http
Generative AI is a revolutionary technology that will revolutionize the way we think about and interact with our lives. It will reshape the way we think about our lives and
```

Setting Different Seed values

On setting a seed value of 42, we get following outputs from the dumb and smart model respectively.

Step 3: Fast Model (`distilgpt2`)

Let's see how the smaller model performs.

```
# Initialize the pipeline with the specific model
fast_generator = pipeline('text-generation', model='distilgpt2')

# Generate text
output_fast = fast_generator(prompt, max_length=50, num_return_sequences=1, truncation=True)
print(output_fast[0]['generated_text'])
```

✓ 11.0s Python

```
Device set to use mps:0
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Both `max_new_tokens` (=256) and `max_length`(=50) seem to have been set. `max_new_tokens` will take precedence. Please refer to the documentation for more information. (http
Generative AI is a revolutionary technology that is now widely used in the field of machine learning.



In the past 15 years, there have been several major announcements on AI that have been made about AI. In the past, there have been several major announcements on AI that have
```

Step 4: Standard Model (`gpt2`)

Now let's try the standard model.

```
smart_generator = pipeline('text-generation', model='gpt2')

output_smart = smart_generator(prompt, max_length=50, num_return_sequences=1, truncation=True)
print(output_smart[0]['generated_text'])
```

✓ 3.3s Python

```
Device set to use mps:0
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Both `max_new_tokens` (=256) and `max_length`(=50) seem to have been set. `max_new_tokens` will take precedence. Please refer to the documentation for more information. (http
Generative AI is a revolutionary technology that will revolutionize the way we think about and interact with our lives. It will reshape the way we think about our lives and
```

On setting a seed value of 48, we get the following outputs from the dumb and smart models respectively.

Step 3: Fast Model (`distilgpt2`)

Let's see how the smaller model performs.

```
# Initialize the pipeline with the specific model
fast_generator = pipeline('text-generation', model='distilgpt2')

# Generate text
output_fast = fast_generator(prompt, max_length=50, num_return_sequences=1, truncation=True)
print(output_fast[0]['generated_text'])
```

✓ 12.9s Python

```
Device set to use mps:0
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Both `max_new_tokens` (=256) and `max_length`(=50) seem to have been set. `max_new_tokens` will take precedence. Please refer to the documentation for more information. (ht
Generative AI is a revolutionary technology that integrates with human technology.
```

Step 4: Standard Model (`gpt2`)

Now let's try the standard model.

```
smart_generator = pipeline('text-generation', model='gpt2')

output_smart = smart_generator(prompt, max_length=50, num_return_sequences=1, truncation=True)
print(output_smart[0]['generated_text'])
```

✓ 9.5s Python

```
Device set to use mps:0
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
Both `max_new_tokens` (=256) and `max_length`(=50) seem to have been set. `max_new_tokens` will take precedence. Please refer to the documentation for more information. (http
Generative AI is a revolutionary technology that, in the past few years, has transformed the way we see and interact with the world, using the "next-generation" AI to automat
```

We can see that the dumb model produces mostly syntactically and semantically correct code but it's not as good of an output as compared to the smart model, which produces a lot more coherent output which is related to the context as well.

Tokenization

In this step, we used the GPT2Tokenizer, as gpt2 is trained on its own special tokenizer, to tokenize a given sentence into tokens and then assigned them IDs which make it easier for the model to make inferences and learn from them.

Let's take a sample sentence.

```
sample_sentence = "Transformers revolutionized NLP."
```

✓ 0.0s

Now we split it into tokens.

```
tokens = tokenizer.tokenize(sample_sentence)
print(f"Tokens: {tokens}")
```

✓ 0.0s

```
Tokens: ['Transform', 'ers', 'Ġrevolution', 'ized', 'ĠN', 'LP', '.']
```

And finally, convert tokens to IDs.

```
token_ids = tokenizer.convert_tokens_to_ids(tokens)
print(f"Token IDs: {token_ids}")
```

✓ 0.0s

```
Token IDs: [41762, 364, 5854, 1143, 399, 19930, 13]
```

POS Tagging

In this step, we tag a given sentence according to the parts of speech it represents, such as nouns, verbs, pronouns, etc.
Some of the POS I have seen in this exercise are,
NNS - Plural Noun
VBD - Past Tense Verb
NNP - Singular Proper Noun
. - Sentence-ending punctuation

Let's tag our sentence.

```
pos_tags = nltk.pos_tag(nltk.word_tokenize(sample_sentence))
print(f"POS Tags: {pos_tags}")
```

✓ 0.1s

```
POS Tags: [('Transformers', 'NNS'), ('revolutionized', 'VBD'), ('NLP', 'NNP'), ('.', '.')]
```

Named Entity Recognition (NER)

In this step, we tag different words in our sentence in order for the model to understand the information they contain and the entity they refer to. This step provides the model with an idea as to whether the word is a Name, an Organization, or a Date and such.

Let's analyze the first paragraph of our text.

Generate + Code + Marko

```
snippet = text[:1000]
entities = ner_pipeline(snippet)

print(f"{'Entity':<20} | {'Type':<10} | {'Score':<5}")
print("-"*45)
for entity in entities:
    if entity['score'] > 0.90:
        print(f"{entity['word']:<20} | {entity['entity_group']:<10} | {entity['score']:.2f}")
```

✓ 0.3s

```
Entity               | Type       | Score
---------------------------------------------
AI                   | MISC       | 0.98
PES University       | ORG        | 0.99
AI                   | MISC       | 0.98
Large Language Models | MISC       | 0.91
LLMs                 | MISC       | 0.90
Transformer          | MISC       | 0.99
```

Project I Built

So the project I built is an AI-Powered Dark Souls Sen's Fortress Game Guide. It is a Question Answering system.

I used the `deepset/roberta-large-squad2` model from the HuggingFace Transformers library, and ran it on my M4 Macbook Air's GPU using the "mps" device setting for accelerated inference.

Roberta-large-squad2 is trained on SQuAD 2.0 dataset, which makes it a much stronger model for question answering, albeit a bit more heavier for the system.

The model references a Game Guide Paragraph containing about 4673 characters. On being asked questions related to the game guide, it is able to answer with a high confidence of ~96%. It is also able to handle batch processing using the `answer_question()` function, and is able to answer 5 questions correctly in a sequence.