GEN-AI

UNIT 1

Lab 1

Name: Anshul Banda

SRN: PES2UG23CS081

Section: B

What is Hugging Face ?

Hugging Face is a forum where multiple models of AI are uploaded and shared for use among engineers.

The models uploaded to Hugging Face are pre-trained. This is done in order to save millions of dollars on training models from scratch and allow users to download models like GPT-2, BERT or RoBERTa directly from Hugging Face for use.

Transformers allow us to access these models by providing us with APIs to download and use these models. It supports framework interoperability, meaning you can often move between PyTorch, TensorFlow, and JAX

Pipelines allow to use these models by abstracting the complex math into three steps:

1. **Preprocessing**: Converts your raw text into numbers (Tokens & IDs) that the model can understand.
2. **Model Inference**: The model processes the numbers and outputs predictions (logits).
3. **Post-processing**: The raw predictions are converted back into human-readable text (labels, answers, summaries).

HANDSON-1_Unit1.ipynb:

The first step we perform in the .ipynb file is to download all the required transformers for the lab.

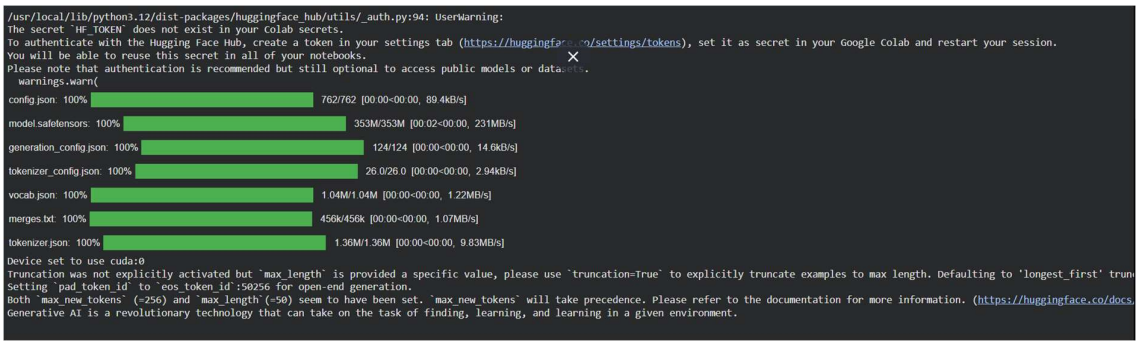
Next we upload the required .txt file for the lab.

First we perform a test between dumb and smart AI models. We compare **distilgpt2** and **gpt2** models.

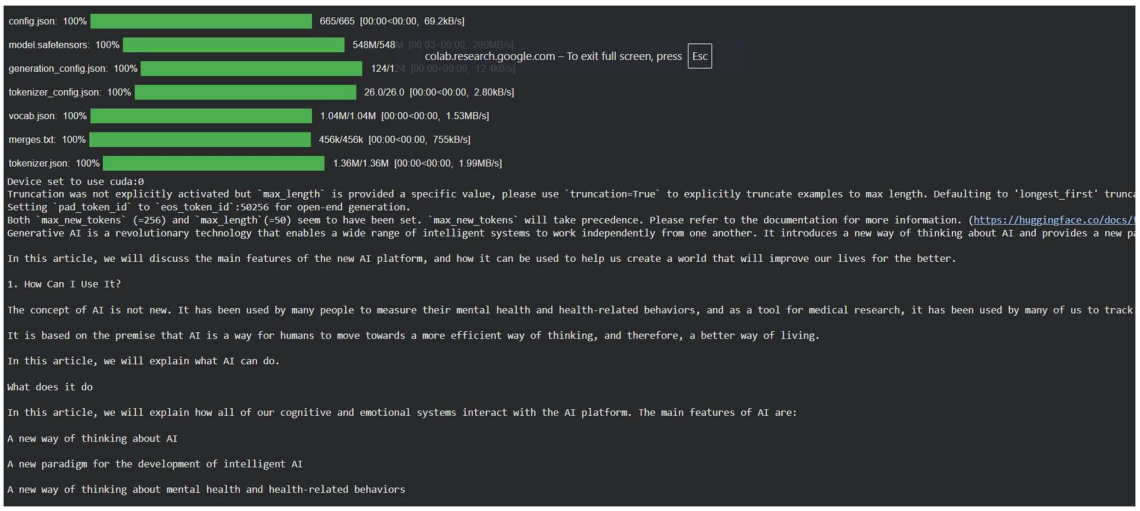
Now we have to set a seed value for the AI models. A seed values sets the randomness of the AI. Same seeds means same random choices which means same outputs, different seeds means different random choices which means different outputs.

For a seed of value 42 and a prompt of “Generative AI is a revolutionary technology that", the AI models produce the current output:

Fast Model (distilgpt2):

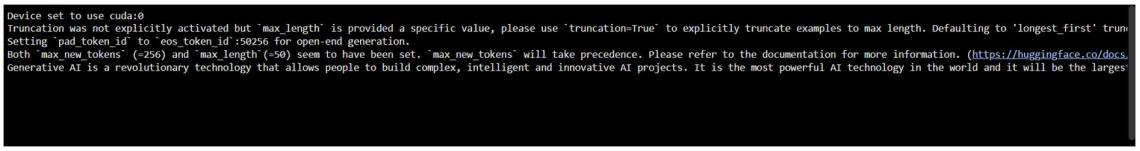


Standard Model (gpt2):

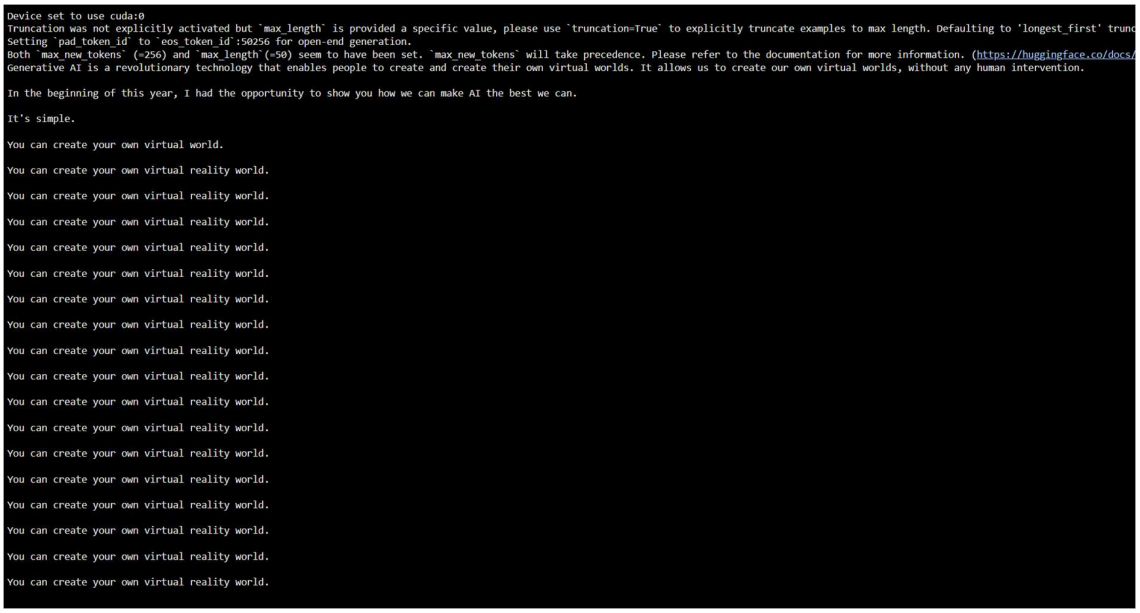


Now changing the seed value to 81:

Fast Model (distilgpt2):



Standard Model (gpt2):



From the two models and the changes in the seed value we can see that the fast model provides us a short but straightforward response to given prompt. However, the standard model gives a longer response to the given prompt, but the standard model also has a tendency to just start drifting into generating nonsensical responses.

NLP Fundamentals: Under the Hood

In order for models to process text, we first need to tokenize them. The pipeline does this for us.

Split a sample sentence "Transformers revolutionized NLP." into tokens.

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

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

And then we assign them IDs:

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

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

PART OF SPEECH(POS):

In order for models to understand the text, it needs to understand context of the text, the words in the text, and how the words in the text as used.

We can see how it recognises verbs, nouns and such:

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

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

Types of part of speech:

NOUNS

NN = Noun, singular or mass

NNS = Noun, plural

NNP = Proper noun, singular

NNPS = Proper noun, plural

PRONOUNS

PRP = Personal pronoun

PRP$ = Possessive pronoun

VERBS

VB = Verb, base form

VBD = Verb, past tense

VBG = Verb, gerund / present participle

VBN = Verb, past participle

VBP = Verb, non-3rd person present

VBZ = Verb, 3rd person singular present

ADJECTIVES

JJ = Adjective

JJR = Adjective, comparative

JJS = Adjective, superlative

ADVERBS

RB = Adverb

RBR = Adverb, comparative

RBS = Adverb, superlative

WRB = Wh-adverb

DETERMINERS / ARTICLES

DT = Determiner

PDT = Predeterminer

WDT = Wh-determiner

CONJUNCTIONS / PREPOSITIONS

CC = Coordinating conjunction

IN = Preposition / subordinating conjunction

NUMBERS

CD = Cardinal number

PARTICLES / SMALL WORDS

RP = Particle

TO = "to"

MODALS

MD = Modal verb

WH-WORDS

WP = Wh-pronoun

WP$ = Possessive wh-pronoun

INTERJECTIONS

UH = Interjection

SYMBOLS / FOREIGN

SYM = Symbol

FW = Foreign word

LS = List item marker

PUNCTUATION

. = Sentence-ending punctuation

, = Comma

: = Colon / semicolon

Summarization: Efficiency vs. Quality

Now we test summarization of different types of models.

We check how well the AI can summarize the output based on this text:
"

The introduction of the Transformer architecture in the 2017 paper "Attention is all you need" was a watershed moment in AI. It provided a more effective and scalable way to handle sequential data like text, replacing older, less efficient methods like recurrence (RNNs) and convolutions.

The fundamental innovation of the Transformer is the attention mechanism. This component allows the model to weigh the importance of different words (tokens) in the input sequence when making a prediction. In essence, for each word it processes, the model can "pay attention" to all other words in the input, helping it understand context, resolve ambiguity, and handle long-range dependencies. This is crucial for tasks like translation, summarization, and question answering.

The Transformer architecture consists of an encoder stack (to process the input) and a decoder stack (to generate the output), both of which heavily utilize multi-head attention and feed-forward networks.

"

Fast Summarizer (**distilbart-cnn-12-6):**

Output:

The introduction of the Transformer architecture in the 2017 paper "Attention is all you need" was a watershed moment in AI . It provided a more effective and scalable way to handle sequential data like text, replacing older, less efficient methods like recurrence (RNNs) and conv

Quality Summarizer (**bart-large-cnn):**

Output:

The introduction of the Transformer architecture in the 2017 paper "Attention is all you need" was a watershed moment in AI. It provided a more effective and scalable way to handle sequential data like text.

Question Answering:

In this part, we will check if the model can answer our questions.

Based on the questions:
"What is the fundamental innovation of the Transformer?",

"What are the risks of using Generative AI?"

The output provided by AI model distilbert-base-cased-distilled-squad is:

```
questions = [
    "What is the fundamental innovation of the Transformer?",
    "What are the risks of using Generative AI?"
]

for q in questions:
    res = qa_pipeline(question=q, context=text[:5000])
    print(f"\nQ: {q}")
    print(f"A: {res['answer']}")
```

```
Q: What is the fundamental innovation of the Transformer?
A: to identify hidden patterns, structures, and relationships within the data

Q: What are the risks of using Generative AI?
A: data privacy, intellectual property, and academic integrity
```

Masked Language Modeling (The 'Fill-in-the-Blank' Game):

In this part, we will see if the AI model can fill in some blanks

```
masked_sentence = "The goal of Generative AI is to create new [MASK]."
preds = mask_filler(masked_sentence)

for p in preds:
    print(f"{p['token_str']}: {p['score']:.2f}")
```

```
applications: 0.06
ideas: 0.05
problems: 0.05
systems: 0.04
information: 0.03
```

From this we observe the probabilities of words the AI thinks are correct for this task.