Intel® GenAI Hackathon

Team Name : TEAM VELOCE

Problem Statement : AI FOR SOCIAL IMPACT

AI FOR SOCIAL IMPACT

Challenge participants to use Stable Diffusion and Transformers to develop AI models that can address social issues. This could involve predicting areas at risk of natural disasters, generating educational content for underprivileged students, etc. The focus here is on open innovation for social good.

UNIQUE IDEA

The Indo-LLaMA tool interprets the text while keeping the essence of what is to be conveyed.

To achieve this the tool scans through {paragraphs → lines} rather than {word → word} interpretation.

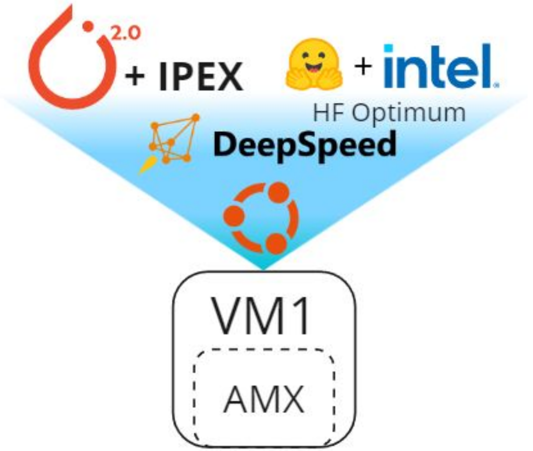
This keeps the meaning intact. The tool also provides a summary of the content provides to get a jist of what is to be conveyed.

ARTIFICIAL RESPONSIBLE INTELLIGENCE?

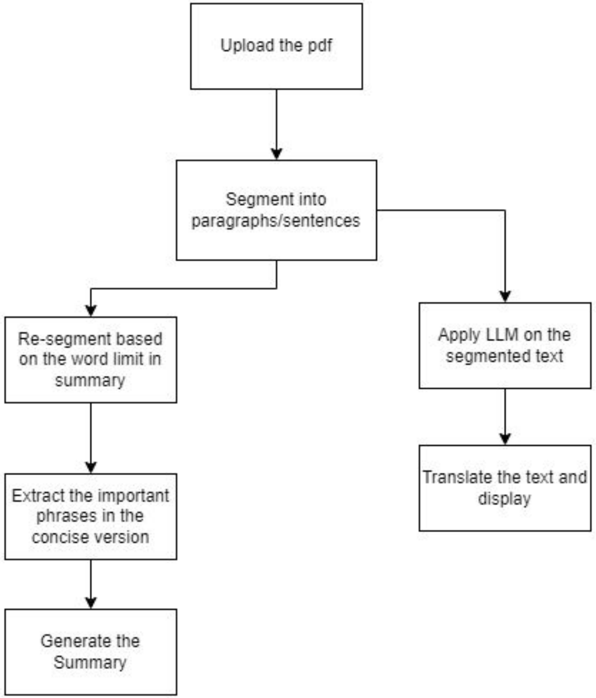
LETS TALK LLAMA

FEATURES OFFERED

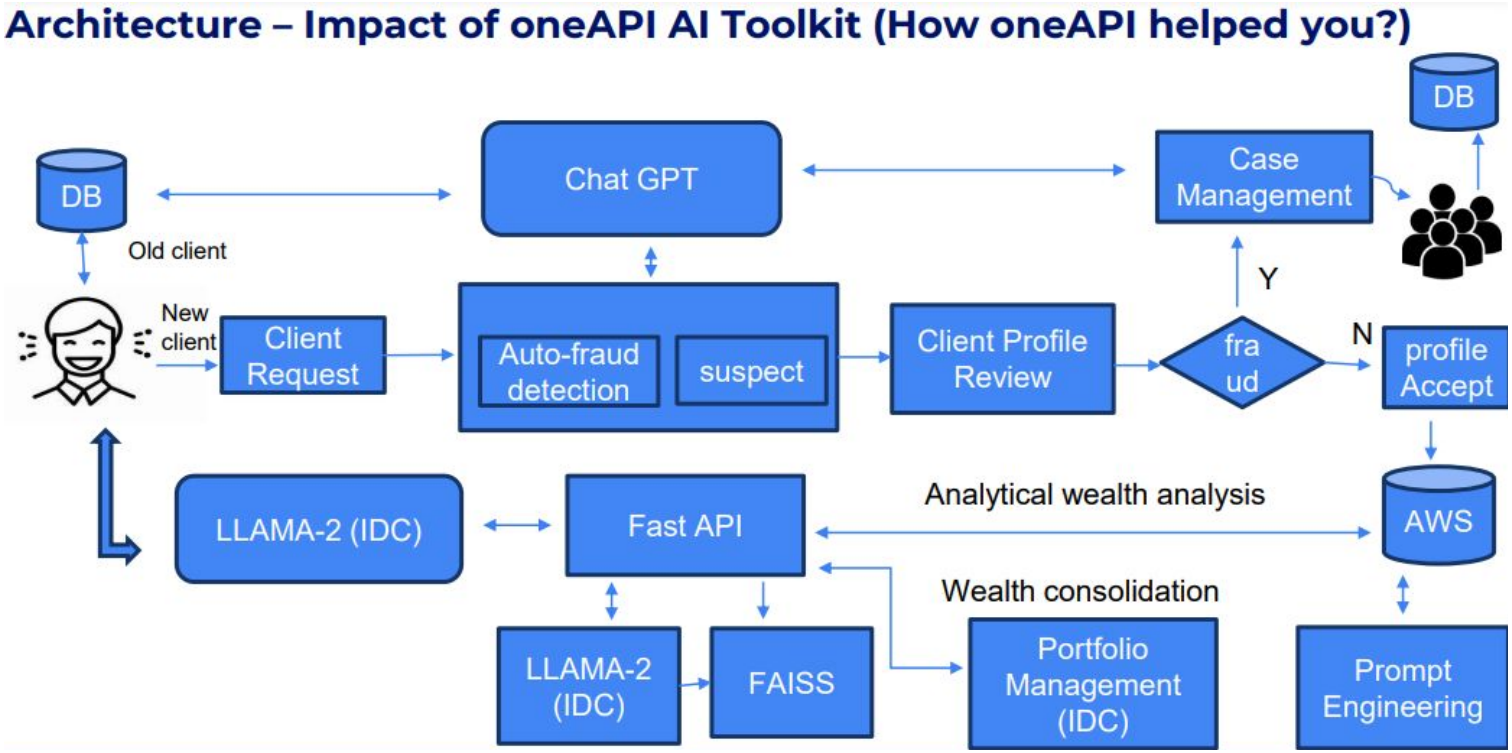
- Translating from and to multiple national/ international languages.
- Inclusion of new languages due to presence of independent tokenizer code
- Provision of a concise summary to understand the gist of the content
- Due to uses of INTEL's Xeon VM, computation time is reduced.
- Large volume of data can be stored on INTEL cloud.



PROCESS FLOW



ARCHITECTURE DIAGRAM



TECHNOLOGIES USED

- *intel-extension-for-pytorch : ipex*
- *Tensorflow*
- *Transformers*
- *deep_translator*

Both pytorch and ipex are used due to dependency issues.

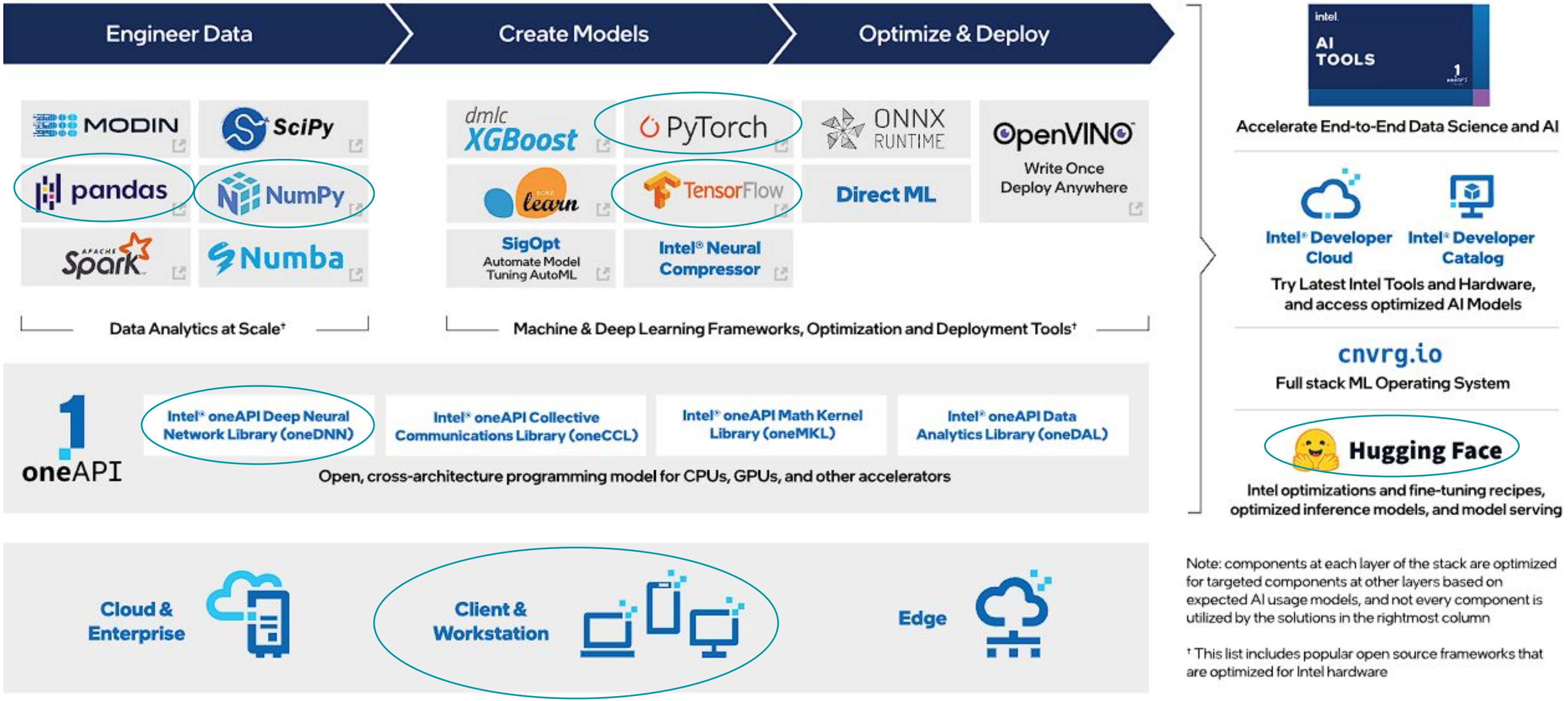
TensorFlow

PYTORCH

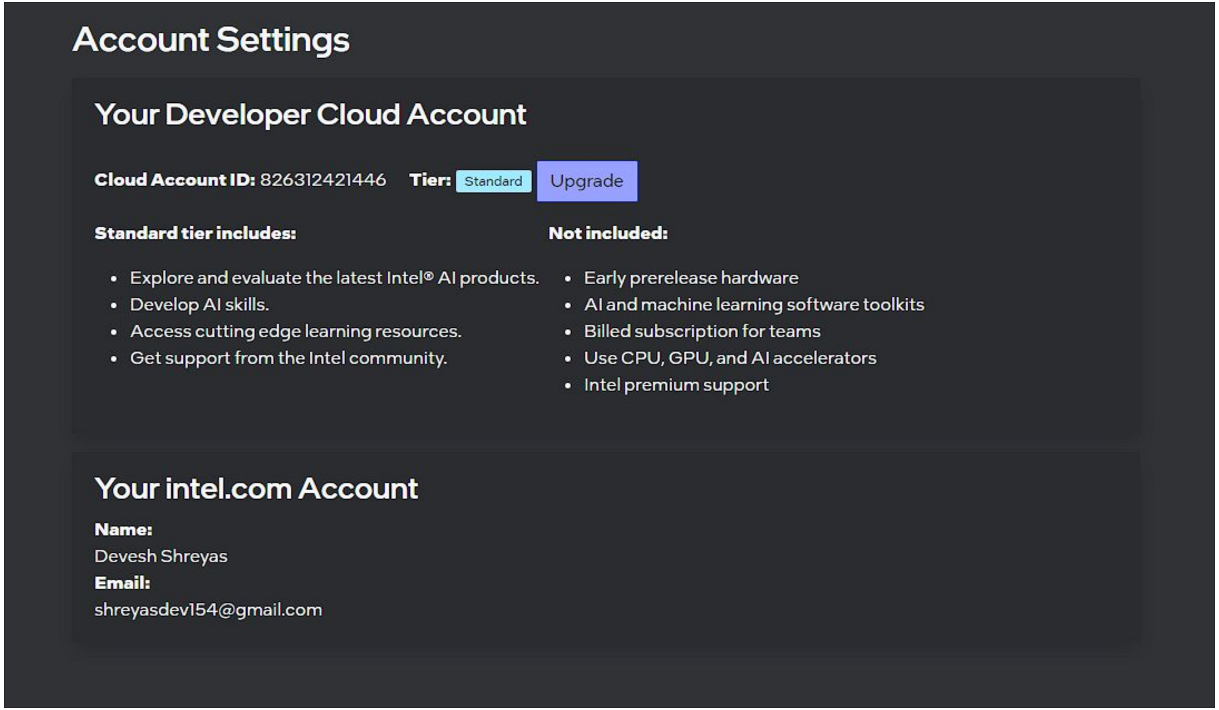
Transformers is a library exclusively for interacting with models on hugging face

At this stage we use these libraries primarily but on development of our own LLM’s with large amount of data, the other tools will come to be useful

SCALABILITY



Intel® Developer Cloud Account (Screenshot)



Use case of Intel® Developer Cloud (IDC)

- **Evaluating and optimizing performance:**
  - IDC provides access to a variety of powerful Intel Xeon processors and GPUs, allowing you to benchmark INDO LLAMA on different hardware configurations and optimize its performance for specific use cases. This can lead to faster processing times and improved efficiency.
- **Scaling development and testing:**
  - IDC's cloud-based nature allows you to easily scale your development and testing environment up or down as needed. This is crucial for handling large datasets or complex text analysis tasks without worrying about hardware limitations.
- **Experimenting with advanced AI frameworks:**
  - IDC offers pre-installed deep learning frameworks like TensorFlow and PyTorch, which INDO LLAMA can leverage for advanced text analysis tasks like sentiment analysis or topic modeling. You can experiment with different frameworks and configurations to find the best fit for your specific needs.

EFFICIENCY AND EFFECTIVENESS BENCHMARK

```
43s [4] model = MBartForConditionalGeneration.from_pretrained("facebook/mbart-large-50-one-to-many-mmt")

pytorch_model.bin: 100% 2.44G/2.44G [00:38<00:00, 32.6MB/s]
/usr/local/lib/python3.10/dist-packages/torch/_utils.py:831: UserWarning: TypedStorage is deprecat
  return self.fget.__get__(instance, owner)()
generation_config.json: 100% 261/261 [00:00<00:00, 14.8kB/s]

4s [5] tokenizer = MBart50TokenizerFast.from_pretrained("facebook/mbart-large-50-one-to-many-mmt", src_

tokenizer_config.json: 100% 528/528 [00:00<00:00, 28.0kB/s]
sentencepiece.bpe.model: 100% 5.07M/5.07M [00:00<00:00, 19.8MB/s]
special_tokens_map.json: 100% 717/717 [00:00<00:00, 42.0kB/s]

1m [10]
    # translate from English to Hindi
    generated_tokens = model.generate(
        **model_inputs,
        forced_bos_token_id=tokenizer.lang_code_to_id["hi_IN"]
    )
```

$T_{total} = 111s$

Run on Google Colab Runtime

In [14]:
```
%%timeit
from transformers import MBartForConditionalGeneration, MBart50TokenizerFast
import torch
from intel_extension_for_pytorch.transformers.optimize import optimize
```

```
1.01 µs ± 118 ns per loop (mean ± std. dev. of 7 runs, 1,000,000 loops each)
```

In [15]:
```
%%timeit
model = MBartForConditionalGeneration.from_pretrained("facebook/mbart-large-50-one-to-many-mmt")
tokenizer = MBart50TokenizerFast.from_pretrained("facebook/mbart-large-50-one-to-many-mmt", src_lang="en_XX")
```

```
2.18 s ± 638 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

In [16]:
```
%%timeit
article_en = "Morality or moral behaviour is not necessarily the result of philosophical reflections. Thesephilosophical reflecti
model_inputs = tokenizer(article_en, return_tensors="pt")
```

```
266 µs ± 1.93 µs per loop (mean ± std. dev. of 7 runs, 1,000 loops each)
```

In [17]:
```
%%timeit
# translate from English to Hindi
generated_tokens = model.generate(
    **model_inputs,
    forced_bos_token_id=tokenizer.lang_code_to_id["hi_IN"]
)
```

```
12.9 s ± 737 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

In [18]:
```
%%timeit
translation = tokenizer.batch_decode(generated_tokens, skip_special_tokens=True)
```

```
71.3 µs ± 13.2 µs per loop (mean ± std. dev. of 7 runs, 10,000 loops each)
```

In [20]:
```
%%timeit
translation
```

```
8.18 ns ± 2.14 ns per loop (mean ± std. dev. of 7 runs, 100,000,000 loops each)
```

Run on Intel Xeon VM and optimised libraries.

$T_{total} = 15s$

$\eta = 86\% \uparrow$

Thank you so much!