**APPROACH**

***Argument Generation using a pre-trained model***

1. In our approach, we use a pre-trained T5 model that is based on the transformer architecture. We therefore don’t have an explicit training step. Instead, we use the already trained parameters for generating conclusions, given an input argument.
2. T5 model is a Text-to-Text Transfer Transformer model being used in our approach for text summarization task.
3. This is an encoder-decoder model that contains an attention layer for contextual awareness while dealing with input sequences. It is pre-trained for different NLP task in the seq2seq problem setting.
4. Text summarization using the T5 model can be done by adding a task-specific prefix to the original input that is fed to the model.
5. The encoder in T5 architecture is trained in a way that BERT is trained, and the decoder is trained in a GPT-style, thereby making it ideal for both text processing as well as text generation.
6. We have used AutoTokenizer with the model type as ‘T5’. The input sequence to the model is encoded into a list of integers with a specific maximum length that is used by the truncation parameter as this parameter has been set to True. The *return\_tensors* parameter is set to ‘tf’ to return tensorflow objects.
7. To generate the summaries in text format, *generate()* function takes care of feeding the encoded input through attention layers to the decoders where the output of the decoder is generated auto-regressively.
8. The maximum length, minimum length and the length penalty of the generated output is set using suitable hyper-parameters along with early stopping.
9. *Decode()* function takes care of converting the list of token integers into a list of strings. At last, we have pre-processed the generated summaries by removing the padding that is usually be added while using T5 model.
10. To summarize, T5 model learns the sequential nature due to its BERT-like architecture, encodes the input sequence using AutoTokenizer and finally decodes the generated output into text format that ultimately performs the task of text summarization.

***Results***

1. We finally created an output JSON file called “*prediction\_out.json*” with predictions by extracting data point IDs from the dataset and generating a dictionary of **ID-prediction** pair.
2. With respect to the run time of our approach, on the local machine, the entire code takes around **10-12 hours** to run to completion.

**EXECUTION INSTRUCTION**

Before proceeding with the execution of the code, it must be ensured that the necessary packages and libraries are present in the system. From the scope of this assignment, the following packages and libraries needs to be installed for successful execution of the assignment:

***numpy, pandas, spacy, json.*** *The* ***TFAutoModelForSeq2SeqLM*** *and* ***AutoTokenizer*** *are imported from* ***transformers*** library*.*

In case any of the above-mentioned packages are missing, it can be installed using the

command **pip install <package-name>.**

In order to download the Spacy “small English language model” used in our code, execute

the command **python -m spacy download en\_core\_web\_sm**

1. Unzip the file Assignment4-HardlyHuman.zip. The unzipped folder will have the following files:
   1. **ArgumentGeneration.ipynb:** Notebook file containing the python code for text summarization task.
   2. **Documentation.pdf**: PDF file describing our approach and execution instructions.
2. To run the code for predicting on the *validation* dataset, click on **Runtime** on the menu bar and click on **Restart and run all** or **Run all**  option directly since the file paths are already set.
3. **If the code must be run** **for predicting on the *test* dataset, the file path must be changed in the code**. This code is present in the first code cell. The value of variable called “*val\_data\_file*” must be changed based on the test data file name. The blue highlight indicates the variable, and the green highlighted part indicates the file name that must be modified. Once the modification is made, select **Restart and run all**.

1. Executing the code, will generate an output file called **prediction\_out.json** with the test/validation dataset.
2. To check the performance of the classifier, download the *prediction\_out.json* file and run the ***eval.py***file from command line/terminal using the following command:

**python eval.py --true <path\_to\_val\_data> --predictions <path\_to\_prediction.out file>**

Running this command will output the **BLUE score** of text generation on test/validation dataset.

Group: Hardly Humans

*Siddhanth Janadri – 6882502*

*Sneha Hiremath – 6881479*

*Sanjay Gupta – 6882964*

*Sai Nikhil Menon - 6882524*