**NLP Project: Text Summarization App**

**Introduction:**
My app, Terms and Conditions Text Summarizer is specifically designed and trained on The purpose of this application is to summarize the Terms and Conditions into an easy-to read format and extract all the key words and relevant information. This app aims to make it more efficient to scan the Terms and Conditions section and decipher the main topics being presented within it. The inspiration behind this application is to make people aware of the agreements and contracts that they sign up for. Oftentimes, many people tend to glance at the terms and conditions without fully reading through it. This is primarily due to how lengthy the Terms and Conditions tend to be. What's more, is that many companies also tend to update their Terms and Conditions periodically, making it difficult for people to keep up with the contracts. Nonetheless it is essential to understand what we're signing up for. We utilize the power of AI and NLP to generate the Term and Condition summaries.

**Usage:**
In order to run this application, copy and paste the "Terms and Conditions" text into the input textbook. Then click the Submit button. This will begin generating the summary and will output it in the left text box entitled "Summary"

**Documentation:**
The frameworks used to build this application include:

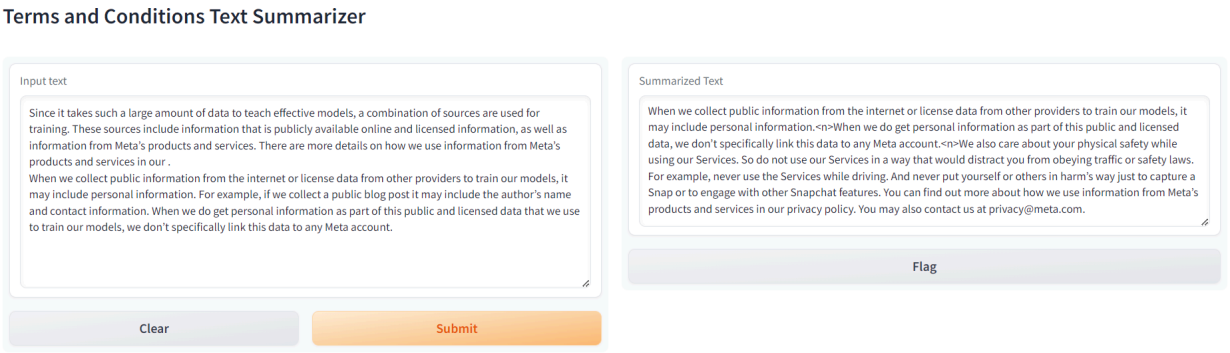
- Hugging Face's Transformer's library, specifically the Summarization pipeline
    - This pipeline makes it very convenient to perform zero shot learning. I utilized this model to gain an insight to the outputs of zero-shot learning and identify areas of enhancements for the model. Since Transformer models are able to also perform text summarization due to their attention blocks, I used this as a baseline to identify the improvements that further refining of the model lead to. I also use this to explore analyze the various methodologies that can help
- Hugging Face's Pegasus Model: (https://huggingface.co/nsi319/legal-pegasus)
    - The reason why I chose this specific Pegasus model is because it's a Transformers-based Abstractive Summarization model and this model has been fine tuned for the legal domain (nsi319/legal-pegasus). Furthermore, the "Terms and Conditions" fall under the legal category and thus will gain more customized training experience when
- RAKE
    - RAKE (Rapid Automatic Keyword Extraction) is a library that automatically calculates the most important words within a corpus based on the frequency and how often it show up alongside other words.This library was used to generate the keywords and most common phrases within the terms and conditions Text. This way, users can glimpse at this list and get a gist of the main topics that the terms and conditions text is focused on.
    - Furthermore, the application also uses RAKE to also retrieve keyword phrases ranked highest to lowest with scores. This helps understand the most essential

keywords within the text and what users should pay attention to within the terms and conditions text.

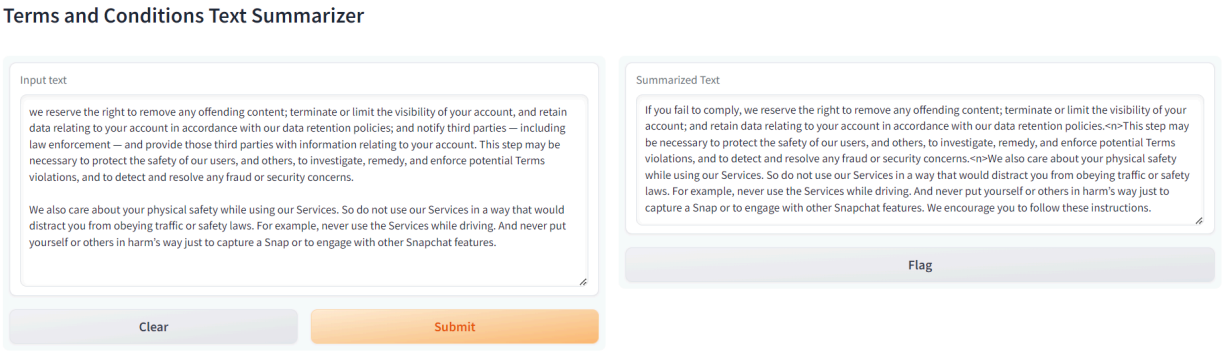
- OpenDatasets (retrieve the dataset from Kaggle that'll be used to finetune the model)
    - Using the OpenDatasets library, I imported the training data that can be used to further fine-tune the model and perform additional training.
- Gradio
    - Used to develop an intractable and user-friendly interface for users to summarize text with the click of a button

Below are sample outputs to my screen:

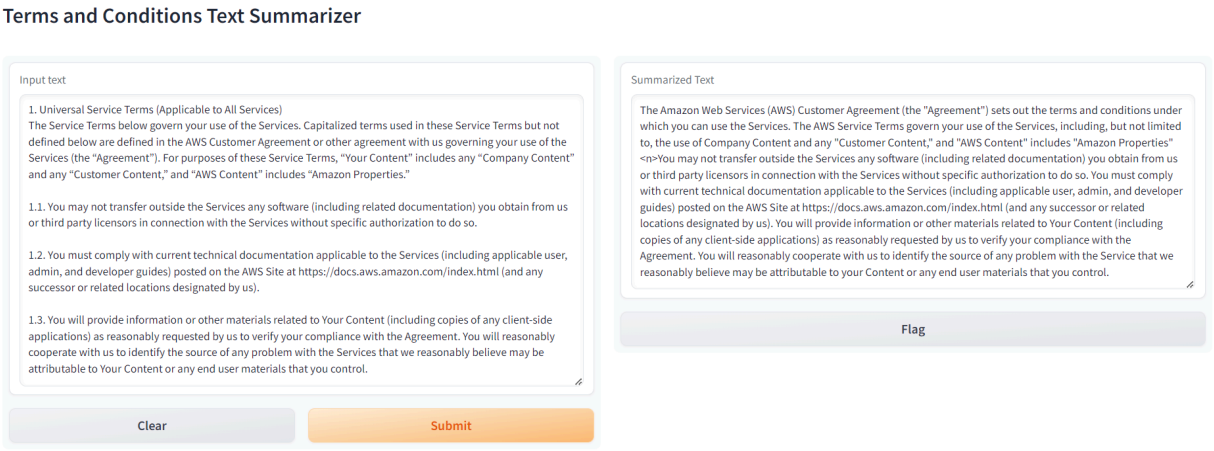
This is a snippet of the summary generated from Instagram's terms and conditions



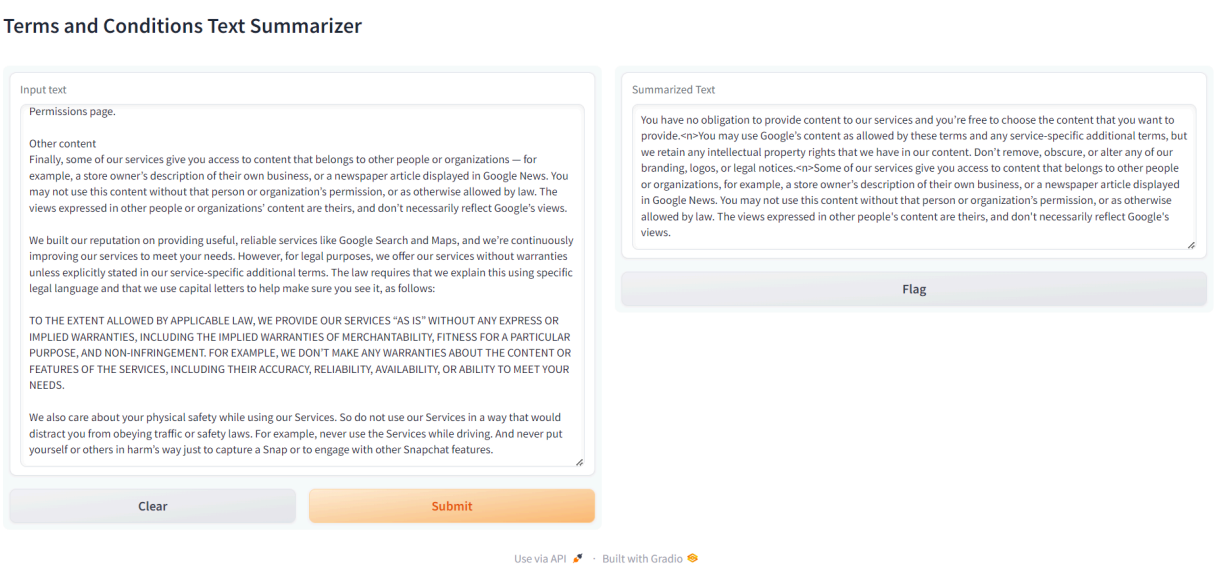

Here is a snippet of the summarization generated from SnapChat's terms and conditions




Here is an example of a larger text retrieved from Amazon AWS's service terms




Finally, here is some summarization from Google's terms of service




**Transformer Model:**

- Our model utilizes the transformer model as well as performing additional finetuning to ensure that the best summaries are generated.

**Contributions:**

In order to develop this model and tailor it specifically for "terms and conditions" I implemented a pre-trained Transformer model, the Pegasus. This Pegasus also came in a fine-tuned format

which was fine-tuned on legal datasets, which is what I ultimately used for my text summarization. Next, I incorporated

My original goal for this project was to start off with a pre-trained model, and then fine-tune the model using the legal text dataset. Then, provide additional training to the model by using a corpus of Terms and Conditions text and their summarized versions and performing sequence classification.

I also explored whether it was possible to further fine tune the already-fine tuned model, I've learned the hard way that using a pre-trained model that has already been fine-tuned and attempting to add another layer of fine-tuning to this required large amounts of GPU and compute power. This resulted in Collab's runtime disconnecting and crashing and I lost all of my session data. To combat this, I tweaked the model and adjusted the number of epochs, batch sizes, length of the input text, etc.  Additionally, I experimented using multiple Transformer models, especially ones which were smaller in than others including distilled-bert, so that the layering did not result in an enormous model. When running on GPU, the model was still struggling to process all of the finetuning and crashed. One of the greatest takeaways from this project was that the be

**Limitations:**
Some of the limitations of this app include size restrictions. Currently the text summarization tool can only process 1024 words at a single time. To combat this, we encourage users to work in batches when entering their Terms and Conditions into our application. Another limitation of the application is that it takes longer time to load when it encounters large portions of text, thus, it is more time consuming to add in a large amount of text at a single time.

As described above, the physical hardware and memory complexities made additional training and enhancements to the model difficult and in most cases impossible. Since this is hosted on Google Colab it has a RAM limit of 32 GB using the free version, I was unable to successfully run my sequence-to-sequence training algorithm. I removed the further fine-tuning implementation using a Terms of Service  dataset  out of my code (so that anyone running my model will not experience any session crashes or failures), however there may still be remnants of it within the text.

Link to the demo:
https://github.com/Khossain01/TxtSummarizationTermsConditions