North South University

CSE499B: Senior Design Project II

Modern Analysis & Design Tools

"AutoNote: Transformative meeting summarization and highlighting points based on NLP. "

Group Members:

Serial	Name	NSU ID	Section
1	Md. Saiyem Raiyan	2012468042	10
2	Sheikh Mohammed Wali Ullah	2021186042	10
3	Samia Sultana	2014048042	23
4	Zobaer Ahammod Zamil	2021796042	10

Models

Pegasus CNN Dailymail:

The previous system generates a summary from the meeting transcript. pegasus-cnn_dailymail model uses the “samsum” dataset where dialogue with summaries exists in that dataset. The hugging API generates the summary also from the transcript. The meeting audio generated a transcript conversion. Then, input the transcript and make the summary of the transcript. The data was collected from a YouTube meeting transcript from audio voice to caption of the subtitle. Therefore, it converted into audio to text. After that, segmentation is the part that divides the data into segments that can be easily analyzed, processed, and converted into text. We fine-tuned the previously selected Pegasus-CNN-Dailymail model. According to these criteria, the model is fine-tuned and trained up to 15 epochs and Tested with 12 fields (paper of Pegasus model) related data. So, in exploratory data analysis, we find a short summary with dataset validation. Besides, it handles problematic transcripts and irrelevant summaries. Our model type is a transformer that learns context and thus meaning by tracking relationships in sequential data like the words in this sentence and evaluating the metrics in rouge scores. To get the best and optimal result, we fine-tuned the model with changing parameters and penalty scores and stratified it with a validation set. Here, we used Stochastic gradient descent (SGD) and normalization. Finally, evaluate the model with performance on validation and testing datasets.

According to our pegasus model, the summary cannot work well. Sometimes, it gives a short summary of the pegasus model, which is good. However, in most cases, it cannot give the meaningful summary that ChatGPT or hugging face API gives. 12 field testing from the pegasus model is good in some cases, but with a huge and long transcript, it cannot give a meaningful summary. So, the model cannot work efficiently. Thus, the summary cannot work well in this format according to the pegasus model. So, by analyzing the new model, we found the EleutherAI/get-neo-2.7 B model, a transformer model designed using EleutherAI's replication of the GPT-3 architecture. GPT-Neo refers to the class of models, while 2.7B represents the number of parameters of this particular pre-trained model. The EleutherAI model works more efficiently than pegasus cnn-dailymail.

Eleuthernet gpt neo 2.7 B & Eleuthernet gpt neo 125 M:

The system generates a summary from the meeting transcript. EleutherAI/gpt-neo model uses the “CNN-DailyMail News Text Summarization” dataset where dialogue with summaries exists in that dataset. Here, we load the model by a transformer model called “Happy Transformer”. We load the model as a generator. It means load with HappyGenerator() function [Code in below]. Firstly, we load this gpt-neo model with 2.7 Billion parameters. It was loaded successfully but unfortunately due to it’s huge size it couldn’t be able to train (cuda memory broke down with 2x T4-GPU and 29 GB RAM). However, for this problem we again try with this model but this time with 125 Million Parameters for training with cnn dataset and

HappyGenerator. Then we tried to train with 1 epoch. It was successfully loaded again and trained [Code snippets and screenshot below]. We tested the train set first with a customized dataset.
At first we merged a 3 column dataset into 1 column dataset. The input text: in the first then summary: like Text <>. Summary: <> So that, the result will be a summary from the given text. Therefore, the training process with 1 epoch was done.
According to EleutherAI/gpt-neo model and happy transformer the summary can not be able to work. This gives redundant output, the same token repeated again and again [screenshot above]. Anyhow it can not generate any good summary. For this major problem we again shift with a new model called “Phi-2” from microsoft.

Microsoft phi 2:

The system generates a summary from the meeting transcript. “Phi-2” model uses the “CNN-DailyMail News Text Summarization” dataset where dialogue with summaries exists in that dataset. The meeting audio generated a transcript conversion. Then, input the transcript and make the summary of the transcript. Firstly, we import a text generation model from the Hugging Face Transformers library and initialize it with a specified model called "microsoft/phi-2". The pipeline function sets up a text generation pipeline with the specified model. Then generates a text summary based on the provided prompt using a text generation model. The summary is presented using Markdown format. Then initializes a tokenizer for the "phi-2" model with a causal language model ("AutoModelForCausalLM") for the "microsoft/phi-2" model, with automatic specification of torch data type and device mapping. Note that the model is pretrain. So, we didn’t train for it. Finally, we generate a short summary of the given prompt using the initialized language model. It utilizes the tokenizer to encode the prompt and the model to generate the summary based on the encoded prompt. The summary is then decoded and printed.
Most importantly, we tried the zero-shot technique here to generate a summary. We also try a few shot techniques for a more accurate summary of a text. It worked pretty well compared to others. After that, from a long meeting, we’ll generate a summary and highlight points.

Mamba:

Linear-Time Sequence Modeling with Selective State Spaces:
The system generates a summary from the meeting transcript. We tried to use the model “Mamba”, the “CNN-DailyMail News Text Summarization” dataset where dialogue with summaries exists in that dataset. But recently, the model has been updated, and after that it is taking a huge amount of time to give a single output.

External Tools

Transformer

The transformer architecture is a critical component of natural language processing (NLP) and other domains. Its deep-learning model and self-attention mechanism effectively capture long-range dependencies in sequential data. The transformer architecture has multiple layers, each consisting of a self-attention mechanism followed by feed-forward neural networks. It is commonly used in encoder-decoder architectures for tasks like machine translation and text summarization. The transformer architecture has revolutionized NLP by providing a powerful and scalable architecture for processing sequential data. This allows for more accurate and efficient modeling of complex relationships within and between sequences.

Pipeline

In the field of natural language processing (NLP), the term "pipeline" refers to a series of processing steps that are applied to input text data in order to achieve a specific NLP task or goal. Each step in the pipeline performs a specific NLP operation, such as tokenization, part-of-speech tagging, named entity recognition, sentiment analysis, or machine translation. NLP pipelines are commonly used to process and analyze text data for various applications, including information extraction, text classification, document clustering, and more.

Happytransformer

HappyTransformer is a Python library that simplifies the use of transformer-based models for natural language processing (NLP) tasks. It offers a high-level interface that can be used across different model architectures and tasks, making it easier to work with transformer models. Some of the key features of HappyTransformer include its unified interface, support for multiple models, ease of use, and open-source nature.

Rouge Score (Analysis)

ROUGE, which stands for Recall-Oriented Understudy for Gisting Evaluation, comprises a collection of metrics utilized to evaluate the excellence of summaries produced by automatic summarization systems. These metrics gauge the degree of similarity between the generated summary and the reference summaries. ROUGE scores serve the purpose of comparing summarization algorithms and tracking advancements in automatic summarization. They are commonly employed in text summarization contests.

BLEU Score (Analysis)

BLEU is a metric used in natural language processing to evaluate machine-generated translations. It measures the similarity between a machine-generated translation and one or more human-produced reference translations based on n-gram overlap. However, BLEU doesn't

consider semantic similarity or fluency, so it's often used alongside other metrics for a more comprehensive evaluation.

Analysis

Context-based analysis for summary (Phi 2, Eleuthernet)

Our supervisor asked us to compare generated summaries with OpenAI summaries, which is considered as ground truth. However, when we tried to compare those, we faced an issue in that there is no summary generated by OpenAI for terrorism-related text. So, we failed to compare these summaries due to the problem of OpenAI.

Again, in Eleuthernet we tried to load the model with happytransformer. But when we loaded the model and tried to generate a summary the model was generating a single token multiple times as an output. For that reason, we didn't get any summary and also couldn't compare the summary with like dataset summary or OpenAI summary.

So, in this case, OpenAI couldn't generate the summary for a specific category of the text but the phi-2 model really saved us this time. Or else, we need to find a new model from hugging face to generate the summary and wait for the comparison to happen.

Length-based (Pegasus)

Our supervisor asked us to check the maximum length of input that is allowed by the Pegasus CNN dailymail model. We tried with a couple of long texts as our meeting script will be so long but found out the model only takes a script with 1200 words or less as an input. This was a problem for the Pegasus CNN dailymail model which was solved when we used Microsoft phi-2 model. Because it can generate output for a very long script. So, Phi-2 model was a lifesaver for us this time.

Token & Temperature (Phi 2)

Our supervisor asked us to compare how the change of maximum token number and temperature affects the generated summary. We tried with different no. of tokens, but at last, we came to a conclusion to use 30% token of the whole text. Moreover, for the temperature, we considered 0.3 as an optimum value because if we change it (raise or lower), we can see that the summary of all types of text is not okay with that. So, in this case, temperature value 0.3 and 30% max token value were the lifesaver for us. Otherwise, we need to check for the whole values range which is a very complicated task.

Platforms & Tools

How did they help?

Hugging face (for choosing models): Hugging Face is a prominent platform in the field of Natural Language Processing (NLP), particularly known for its contributions to NLP research and its library of pre-trained models like Transformers. Initially, hugging face helped us to find a lot of models regarding this kind of NLP summarization projects. Additionally, by collaborating with this lot of NLP models embedded here like BERT, Roberta, Transformer Models etc. As a result, it has become easy for us to pick models and effective models. From the models we achieve a lot of technological knowledge, research papers ideas, model architecture, codes and manuscript ideas.

Besides these ways hugging face helped us to find many things like:

Transformers Library: We can be able to experiment with different model architectures and hyperparameters (tuning knobs) to customize the summarization process. So, we can utilize pre-trained models like BERT, GPT, or T5 for tasks related to meeting summarization.

Fine-Tuning: Hugging Face provides tools and resources for fine-tuning pre-trained models on specific tasks, allowing us to adapt them to our particular needs. For our project, we tried to fine-tune a summarization model on meeting transcripts or documents to generate relevant summaries. Again, we used our new models and tried to fine-tune them.

Model Hub: We can explore different models and see which ones perform best for our specific task. That's how we move forward by analyzing. This can save our time by not having to train models from scratch.

Datasets: The Hub provides datasets like SAMSUM Dataset, DIALOGSUM Dataset, XSUM Dataset etc that can be used to train or evaluate our models.

Community Support: Hugging Face has a large and active community of NLP practitioners and researchers. We can engage with this community through forums, GitHub repositories, and social media channels to seek advice, share insights, and collaborate on projects.

Documentation and Tutorials: Hugging Face provides extensive documentation and tutorials to help users get started with their platform and libraries. These resources can be invaluable for understanding how to use their tools effectively.

Colaboratory Google:

Google Colaboratory (Colab) is a popular platform for running Python code, particularly for machine learning and data analysis tasks.

Free Access to GPU and TPU: Colab provides free access to GPU and TPU resources, which can significantly speed up training for NLP models, especially large ones like transformers. This is crucial for experimenting with various models and tuning hyperparameters efficiently without worrying about hardware constraints.

Pre-installed Libraries: Colab comes with pre-installed popular libraries such as TensorFlow, PyTorch, and scikit-learn, which are commonly used in NLP projects. This saves time in setting up the environment.
Collaboration Features: Colab allows for easy sharing and collaboration on notebooks, making it easy for teams working on the same project. So, Multiple users (we) can work on the same notebook simultaneously, facilitating collaboration and knowledge sharing.
Integration with Google Drive: Colab integrates seamlessly with Google Drive, allowing us to access and save notebooks directly to our personal drive. This ensures that our work is automatically saved and can be accessed from anywhere.
Resource Management: Colab provides resources on a temporary basis, which may not be suitable for long-running tasks. However, we connected Colab with our Google Drive and saved checkpoints to resume training later.
Documentation and Community Support: Colab has extensive documentation and a large community of users who share tips, tricks, and solutions to common problems.
Transformers: This library includes pre-trained NLP models like BERT and RoBERTa, which can be fine-tuned for meeting summarization tasks. For example: we used a happy transformer by using the Eleuthernet gpt neo model for generating summary.

Kaggle:

Kaggle is a platform primarily known for hosting machine learning competitions, datasets, and kernels (code notebooks).
Datasets: Kaggle hosts a vast repository of datasets, which could include meeting transcripts, summaries, or related data that we used for training and evaluation of our models. Searching for relevant datasets like cnn dataset can save our time and effort in data collection.
Notebooks: Several notebooks on Kaggle deal with meeting transcript summarization. These notebooks showcase code for data cleaning, pre-processing, and summarization techniques we learn from. Here, we used to load and train our models very easily because it has a lot of space of 2x T4 GPU, 2x 15 GB RAM, 73 GB Hard Disk. So it is easier to work on that.
Kernels: We found kernels related to NLP tasks, including summarization, text generation, and more. These kernels serve as valuable resources for learning from others' code, experimenting with different techniques, and even building upon existing solutions.
Community: Kaggle has a large and active community of NLP practitioners. We engaged with this community through forums, discussions. Asking questions, seeking feedback, or sharing our progress can provide valuable insights and support for our project.

OpenAI:

OpenAI is the most crucial and helpful part of this project. NLP meeting summarization is the most challenging task in the modern world. Lots of models can be used through open source AI which is the only way to help users. Unfortunately, this doesn't have any good resources or a large community. So, it has become difficult for us to deal with this major challenge. The only

optimal way was the OPENAI that we used. This can help us to grab a lot of knowledge regarding NLP and its models. Additionally, this gave us a lot of solutions to load models, train models, debugging, solve errors etc. Mostly, we used it for ground truth and possible solutions. Therefore, it becomes easier for us to work with that. We used ChatGPT, Gemini, sometimes google colab. Without OPEN AI no other challenging task can be solved smoothly.
Here's some other advantages of this:

Scalability for Real-Time Processing: This scalability ensures that our system can handle meetings of varying sizes and durations efficiently.

Customization for Specific Needs: OpenAI's APIs offer flexibility and customization options, enabling us to tailor the summarization and highlighting process to meet the specific requirements of our project.

Integration with Existing Tools: OpenAI's APIs can be seamlessly integrated with existing tools and applications used for meeting management and collaboration.

Community Support and Resources: Leveraging this community can provide valuable guidance and assistance throughout the development process.

Jupyter Notebook/Anaconda:

Jupyter Notebook is excellent for iterative development and experimentation. It supports various programming languages, including Python, which is widely used in NLP.

Interactive environment: Jupyter Notebook allows us to write, execute, and visualize our code in a single interface. This is particularly useful for experimenting with different NLP techniques and visualizing our results as we go.

Readability: Jupyter Notebooks combine code, text, and visualizations into a single document, making your project more readable and reproducible for ourselves and others.

Sharing: Jupyter Notebooks can be easily shared with collaborators, allowing for seamless teamwork on our project.

Package management: Anaconda simplifies the process of installing and managing Python libraries and packages we'll need for our project, including popular NLP libraries like NLTK, spaCy, and TensorFlow.

Environments: Anaconda allows you to create and manage virtual environments for our projects.

How did we find them?

Hugging face:

Supervisor: Our supervisor of this project helped us to find models, datasets by suggesting this "**Hugging face**" Platform.

Search: The Hub's search function is a great way to find models, datasets, and pipelines based on keywords like "meeting summarization," "NLP," or specific model names.

Community Discussions: Explore discussions and forums on the Hugging Face platform to learn from other NLP enthusiasts and get insights on model selection and usage.

Colab:

For Finding Libraries:

Search: We search for libraries directly within Colab using the !pip install <library_name> command.

Community Notebooks: Look for publicly shared Colab notebooks on GitHub or Kaggle that tackle similar NLP tasks. These notebooks often include pre-installed libraries and code snippets we adapted. This helps us a lot.

Kaggle:

In terms of finding these resources on Kaggle, we started by browsing the datasets and competitions sections and searching for relevant keywords related to our project. Additionally, exploring kernels related to NLP and summarization tasks can provide valuable code examples and techniques. We found our most valuable dataset from here so that we worked so smoothly by splitting the dataset.

OpenAI:

- Explore OpenAI's website and documentation.
- Read blog posts and case studies for real-world insights.
- Join developer communities and forums for support.
- Experiment with sample code and projects on platforms like GitHub.
- Reach out to OpenAI's support team or community for assistance.

Jupyter Notebook/Anaconda:

Start with official documentation and community forums for guidance and support. Explore YouTube tutorials, online courses, and books dedicated to data science and NLP, which often cover practical implementations. Additionally, GitHub repositories, blogs, and LinkedIn groups offer valuable insights, code examples, and discussions to enhance our understanding and productivity with these tools.

Why is that not helpful?

Hugging face:

Models with Limited Documentation or Performance: Some models might have limited documentation or unclear training details, making it difficult to understand their suitability for our task. We need to experiment and evaluate different models.

Complexity of Fine-Tuning: While fine-tuning pre-trained models can be incredibly powerful, it may also be challenging for us because we are not familiar with those concepts or model training. Fine-tuning requires understanding concepts like hyperparameter tuning, data preprocessing, and model evaluation, which can be daunting for us.

Resource Intensiveness: Training or fine-tuning large models can be computationally intensive and may require significant computational resources, including GPUs or TPUs.

Model Selection: With the abundance of models available on the Hugging Face Model Hub, selecting the most appropriate one for a specific task can sometimes be overwhelming. It requires understanding the different architectures, hyperparameters, and training data.

Community Noise: While community support can be invaluable, sorting through forum threads or GitHub issues to find relevant information or solutions to specific problems can sometimes be time-consuming, especially if there's a lot of noise or unrelated discussions.

Colaboratory Google:

Limited Resources: While Colab offers free GPU and TPU resources, they are limited and may not be sufficient for large-scale training or processing tasks. Additionally, there are session time and idle timeouts, which can interrupt us for long-running processes.

Dependency on Internet Connection: Colab requires an internet connection to access and run notebooks, which can be a limitation if we need to work offline or have unreliable internet access.

Data Privacy Concerns: Since Colab notebooks are hosted on Google servers, there are concerns about data privacy and security, especially if working with sensitive data.

Outdated and Unmaintained Libraries are another issue thus creating drawbacks.

Kaggle:

Kaggle took a lot of time to train and if the internet somehow goes down for a while the kernel or session become stopped. Thus the model needs to load again from the beginning which is time consuming. On the other hand, the dataset was not fully customized so that we again customized the train, test and validation set. While some kernels may provide valuable insights and solutions, others might not be as relevant or well-documented.

OpenAI:

- General-purpose AI resources not specifically tailored to NLP or meeting summarization.
- Outdated or irrelevant documentation that doesn't address our project's needs.
- Communities or forums focused on different areas of AI or unrelated topics.
- Sample code or projects that are not applicable and don't provide insights relevant to our specific project.
- Sometimes lack of responsive support and limited availability.

Which ones were life savers and which ones were a waste of time?

Hugging face:

Life savers:

- Pre-trained models from Hugging Face's Transformers library.
- Community support for troubleshooting and sharing insights.
- Access to a wide variety of pre-trained models via the Model Hub.

Waste of time:

- Spending too much time on complex fine-tuning without clear goals.
- Spending training in local server and big models in kaggle took a lot of time.
- Misguided model selection without understanding their suitability.
- Over-reliance on documentation without practical experimentation or community engagement.

Colaboratory Google:

Life savers:

- Free access to GPU and TPU resources.
- Pre-installed libraries like TensorFlow and PyTorch.
- Collaboration features for team projects.
- Integration with Google Drive for easy storage and access.
- Extensive documentation and community support.

Waste of time:

- Limited resources for long-running or large-scale tasks.
- Dependency on internet connection for access.
- Potential data privacy concerns with hosted notebooks.

Kaggle:

Life savers:

Relevant datasets: Access to diverse datasets for training and evaluation.
Kernels: Code examples and techniques for NLP tasks, providing valuable insights and solutions.
Community: Engagement with a supportive community for feedback, discussions, and learning opportunities.

Waste of time:

Irrelevant competitions: Competitions not directly aligned with our project may not offer substantial value.
Unhelpful kernels: Some kernels may lack relevance, quality, or documentation, making them less useful for learning or implementation.
Session: The session become stopped due to internet issues. So from the beginning it wastes a lot of time and for model and dataset loading.

Jupyter Notebook/Anaconda:

Life savers:

This typically includes comprehensive documentation, active community forums, and well-structured tutorials that offer practical guidance and solutions to common challenges. These resources can significantly expedite the learning process and aid in overcoming obstacles during project development.

Waste of time:

As for wastes of time, they could encompass overly complex tutorials, outdated courses, or low-quality content lacking practical relevance. It's essential to discern between valuable resources and those that may not contribute meaningfully to our project goals, ensuring our time and efforts are efficiently utilized.