Text Summarization using BiLSTM, Pegasus, T5-small using Langchain

Pooja Bejjanki, Pavani Daggula and Jerusha Bhaskaran

1. **INTRODUCTION**

The exponential growth of textual data has created a strong need for efficient summarization techniques to extract meaningful insights. In a world where new information is constantly being shared it is vital to filter through content fast. This is where the natural language processing task, text summarization, comes into play. Text summarization can be categorized in two ways. Extractive summarization; A method selects key sentences directly from the text without changing them. The second is abstractive summarization which generates new sentences that summarize the text in human-like summaries [12]. Abstractive summarization is flexible and natural but also more complex. During our research, we found that some models were referenced multiple times. Traditional models like long short-term memory (LSTM), and BiLSTM, have successfully understood sequential patterns in text. Newer transformer-based models like Pegasus and T5 use advanced techniques, such as self-attention mechanisms, to capture the context better and create more meaningful summaries [16]. In our research, LSTM had multiple variants in which we chose BiLSTM due to its ability to account for preceding and subsequent context [2]. Langchain offered unique features as a framework such as strengthening the model it is working around as well as its prompt engineering [5]. Pegasus was also found in multiple papers when looking for models that work with natural language processing.

This project compares three summarization models: BiLSTM, a traditional sequence-to-sequence model that processes input text in both forward and backward directions; Pegasus, a transformer model specialized in abstractive summarization; and T5 small, a lightweight transformer model designed for text-to-text tasks. The goal is to evaluate each model’s summarization method based on output using BERTscore.

1. **DATASET**

*Hugging Face SAMSum Corpus*

Hugging face library contains a number of datasets that can be used for natural language processing tasks. Our group decided to use the SamSum dataset, a corpus containing dialogues written in English paired with summaries. The corpus contains a total of sixteen thousand three hundred and sixty nine instances [1].Each instance has three fields. The first column contains dialogue, the second has a summary and the third column has an unique ID of each instance as shown below. The entire corpus is split into ninety percent training, five percent validation and five percent testing data. There are a number of topics creating a diverse range for the models to train on. Human linguists wrote the summaries. It was not machine generated.

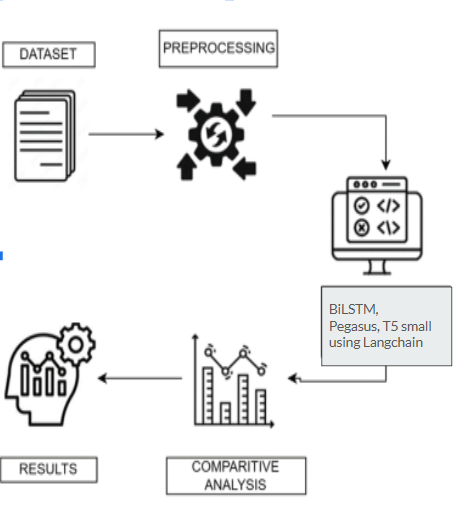
Split lengths: [14732, 819, 818]

Features: ['id', 'dialogue', 'summary']

| id | dialogue | summary |
| --- | --- | --- |
| 13729565 | Eric: MACHINE!  Rob: That's so gr8!  Eric: I know! And shows how Americans see Russian ;)  Rob: And it's really funny!  Eric: I know! I especially like the train part!  Rob: Hahaha! No one talks to the machine like that!  Eric: Is this his only stand-up?  Rob: Idk. I'll check.  Eric: Sure.  Rob: Turns out no! There are some of his stand-ups on youtube.  Eric: Gr8! I'll watch them now!  Rob: Me too!  Eric: MACHINE!  Rob: MACHINE!  Eric: TTYL?  Rob: Sure :) | Eric and Rob are going to watch a stand-up on youtube. |

1. **METHODOLOGY** 
   1. *Data Preprocessing*

We removed special characters and excessive spaces from the dataset. We also got rid of the unique identifier as it was not necessary for our project.

****

*Figure 1: Methodology*

B. *Models*

1. *Long Short-Term Memory*

*Why Bi-LSTM Model*

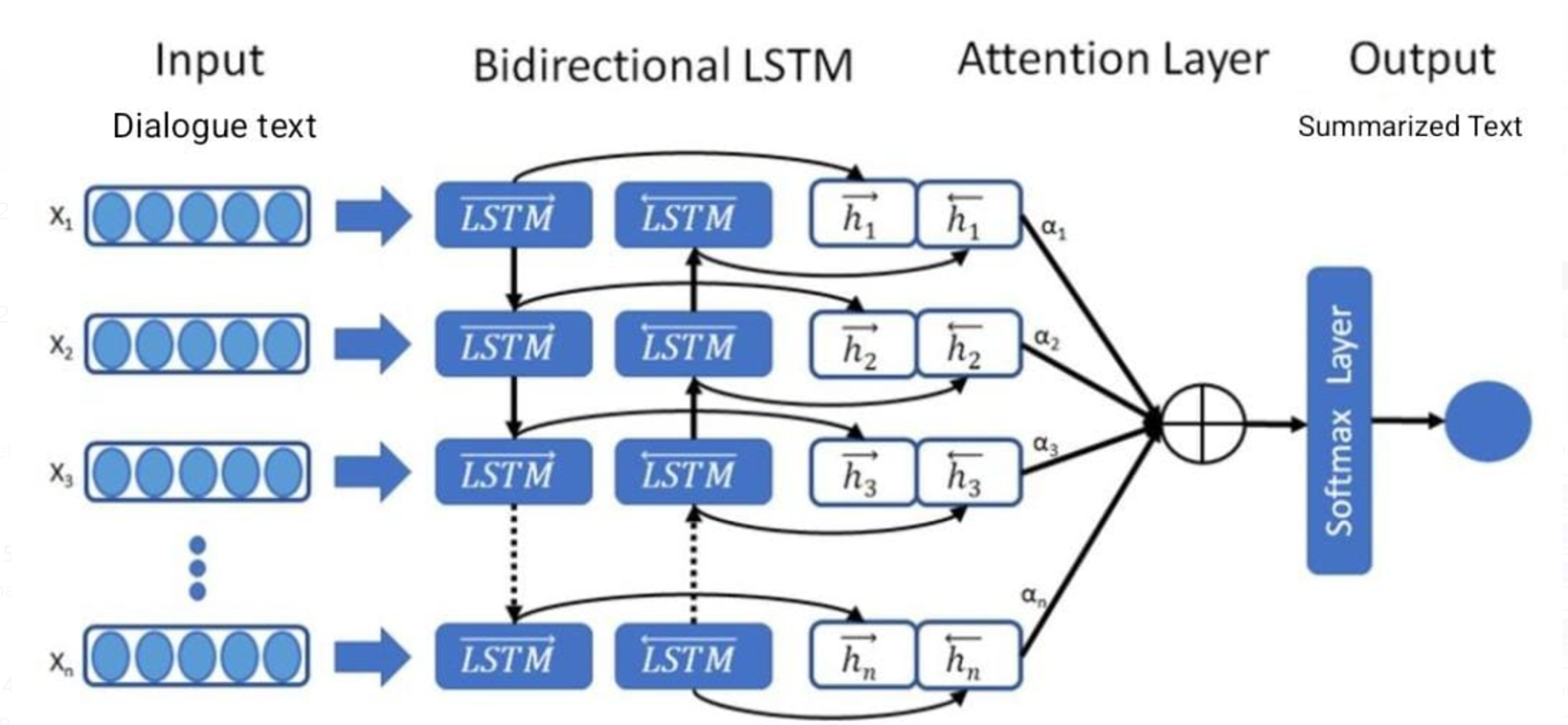
*Sequential Data Handling:* Bi-LSTM effectively captures context from both past and future words in a sequence. This bidirectional processing makes it ideal for dialogue summarization, where understanding the full context is crucial for generating accurate summaries[3].

*Preprocessing Flexibility:* The Bi-LSTM model supports extensive preprocessing techniques and embedding-based inputs. This flexibility enables the integration of cleaned and tokenized conversational data, ensuring better representation and model performance[4].

*Architecture*

*1)Input Layer*The input to the model consists of tokenized and padded sequences derived from the Samsum dataset. Tokenization involves splitting text into words and assigning unique integer indices, while padding ensures all sequences have the same length for uniformity.

*2)Embedding Layer*The tokenized input is passed through an embedding layer, which converts each word index into a dense vector of fixed dimensions. These embeddings capture semantic relationships between words, helping the model understand the context of the text.

**

*Figure 2: LSTM Architecture*

*3)Bi-LSTM Layer*The core of the model is a Bidirectional LSTM layer, which processes the sequence in both forward and backward directions. This allows the model to capture contextual information from both past (previous words) and future (subsequent words) contexts in the dialogue, improving its ability to summarize conversations accurately.

*4)Dense Output Layer*The output of the Bi-LSTM layer is fed into a dense (fully connected) layer, which generates the final output. For this task, the dense layer predicts the sequence of words that form the summary.

*Hyper parameters*

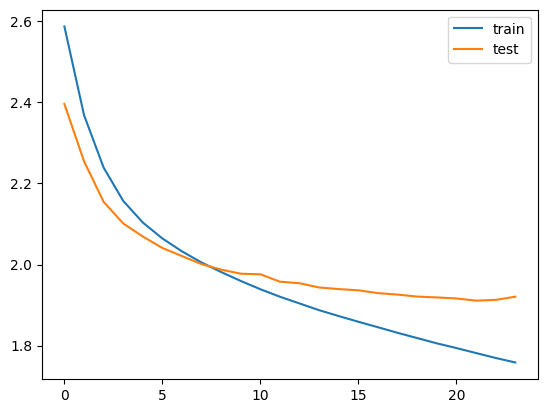
Model Compilation

* Optimizer: rmsprop
* Loss Function: sparse\_categorical\_crossentropy

Training Parameters

* Hyper parameters are selected by trial and error.
* Epochs: 30 (The model is trained for 30 iterations over the entire dataset)
* Batch Size: 25 (The number of samples processed before the model updates its parameters)

*Loss curve*

**

*Figure 3: Loss curve for BiLSTM*

* The training and test loss curves consistently decrease, indicating effective learning by the Bi-LSTM model and proper optimization.
* The minimal gap between the training and test loss suggests the model is not overfitting and is capable of generalizing well to unseen data. This highlights the effectiveness of techniques like dropout regularization used during training
* The curves stabilize around epoch 20, suggesting the model achieves sufficient learning without additional training epochs.
* The trends confirm the model's capability to learn meaningful dialogue representations for accurate summarization.

Output:



*Figure 4: Output of BiLSTM for test data*

Challenges

*Handling Long Sequences*

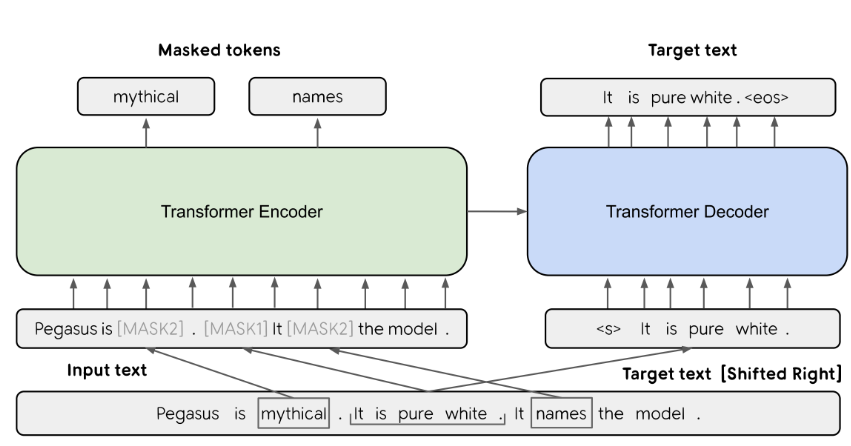
Dialogue datasets, like the Samsum dataset, often contain long sequences. Bi-LSTM models may struggle with computational efficiency and memory requirements when processing these sequences, leading to increased training time and the risk of vanishing gradients.

*Limited Contextual Understanding*

While Bi-LSTMs capture context bidirectionally, they may still miss nuances in highly complex or ambiguous dialogues, especially when the dataset contains diverse conversational patterns.

*Hyperparameter Tuning*

Choosing the right hyperparameters, such as the number of LSTM units, embedding dimensions, learning rate, and dropout rate, is non-trivial and often requires extensive experimentation.

*B. Pegasus*

Pegasus is a transformer based encoder decoder architecture designed explicitly for abstract summarization[7]. It Leverages a novel pre-training objective known as gap sentence generation (GSG), which trains the model to reconstruct sentences from a document. This document enables Pegasus to understand the global context of a text and generate coherent and concise summaries. Pegasus excels in abstractive summarization tasks because it generates new sentences that effectively encapsulate the source text[7]. It handles long and complex documents with ease, making it suitable for varied domains like articles and documents *Figure 5: Pegasus Architecture*   
*Implementation Details:*

* The pre-trained google/pegasus-cnn\_dailymail model was fine-tuned on the SAMsum dataset to adapt to conversational text summarization.
* The Autotokenizer ensures compatibility during text preprocessing. The training utilized the Hugging Face Trainer API with following configuration:  
  Epochs, batch size (tried with different epochs ), Gradient accumulation, optimizer,max\_length, length\_penalty were fine tuned during inference for primal performance.

*Evaluation:*

Here is the example summary generated by Pegasus

| dialogue | Reference Summary | Model Summary |
| --- | --- | --- |
| Hannah: Hey, do you have Betty's number?  Amanda: Lemme check  Hannah: <file\_gif>  Amanda: Sorry, can't find it.  Amanda: Ask Larry  Amanda: He called her last time we were at the park together  Hannah: I don't know him well  Hannah: <file\_gif>  Amanda: Don't be shy, he's very nice  Hannah: If you say so..  Hannah: I'd rather you texted him  Amanda: Just text him 🙂  Hannah: Urgh.. Alright  Hannah: Bye  Amanda: Bye bye | Hannah needs Betty's number but Amanda doesn't have it. She needs to contact Larry. | Amanda: Ask Larry Amanda: He called her last time we were at the park together .<n>Hannah: I'd rather you texted him .<n>Amanda: Just text him . |

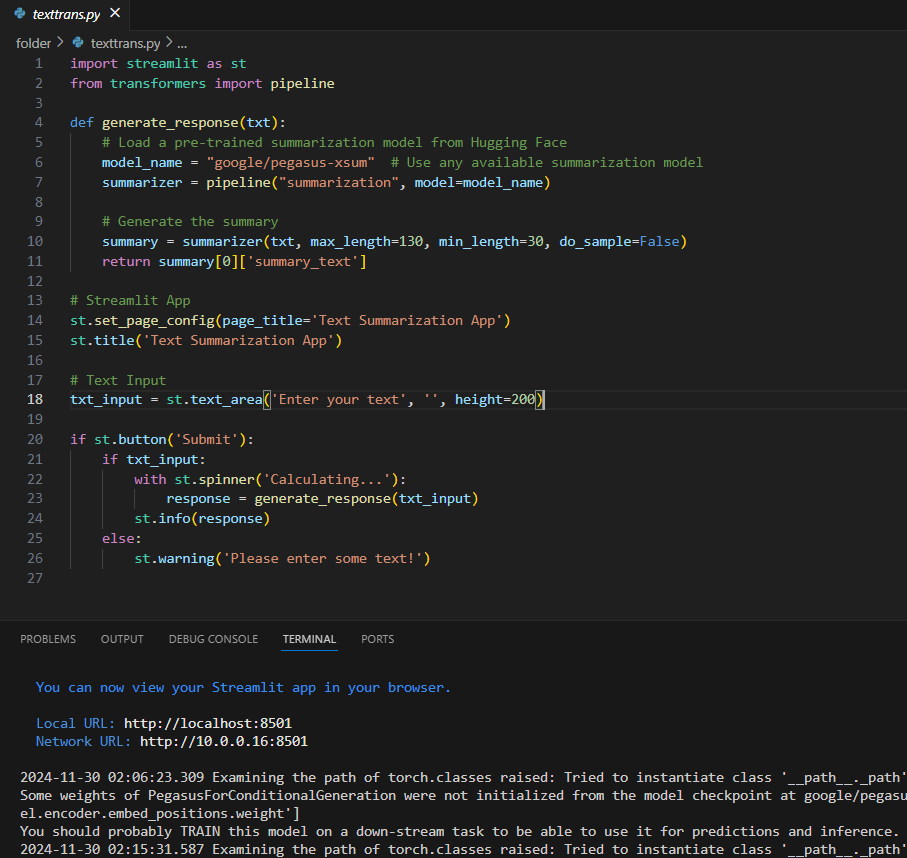
*BERTScore Results:*

* *Precision: 0.8003*
* *Recall: 0.8939*
* *F1 Score: 0.8444*

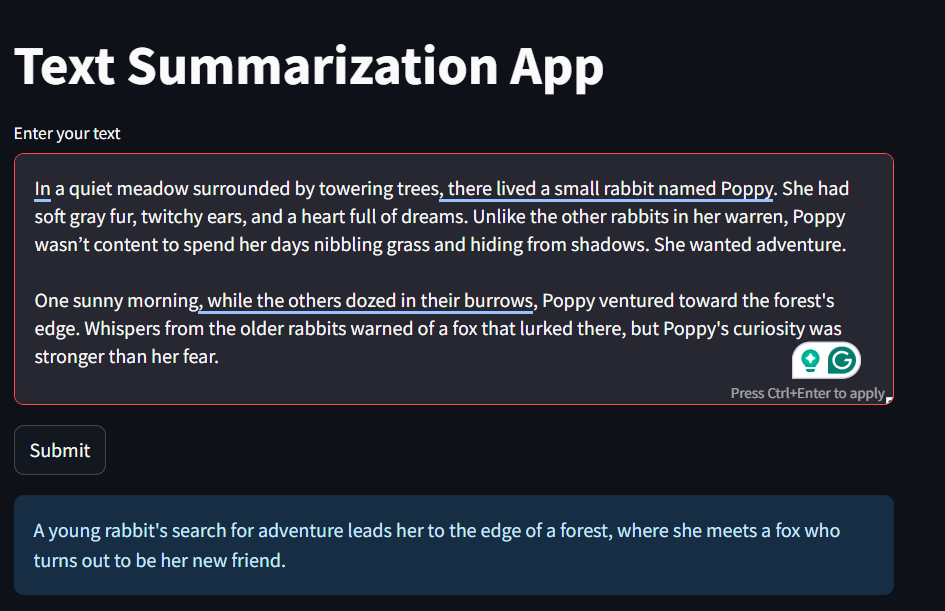
High recall indicates that Pegasus is particularly effective at retaining essential information from input dialogues and F1 score is 0.84 this indicates Pegasus captures key details effectively and produces rich, semantically accurate summaries, especially for complex dialogues.

*Text Summarization tool:*

A Streamlit based web application was developed to showcase the implementation of pegasus for text summarization. The app uses the pegasus pretrained model to generate real time summaries from user provided text. The screenshot is attached for reference below[8].



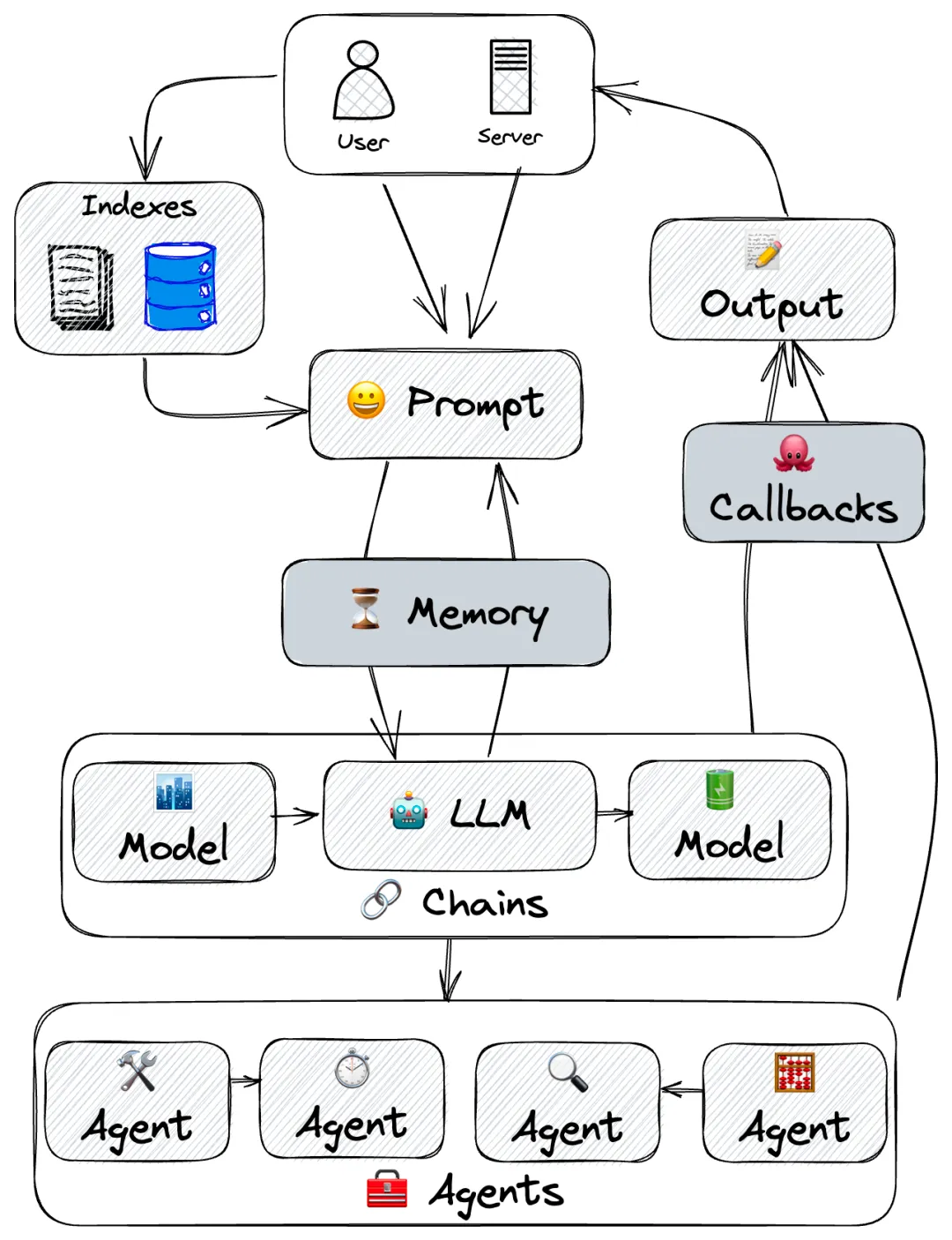
*Figure 6:Code snippet for text summarization App*



*Figure 7: Application overview*

**C.** *T5 Small with LangChain Framework Langchain*

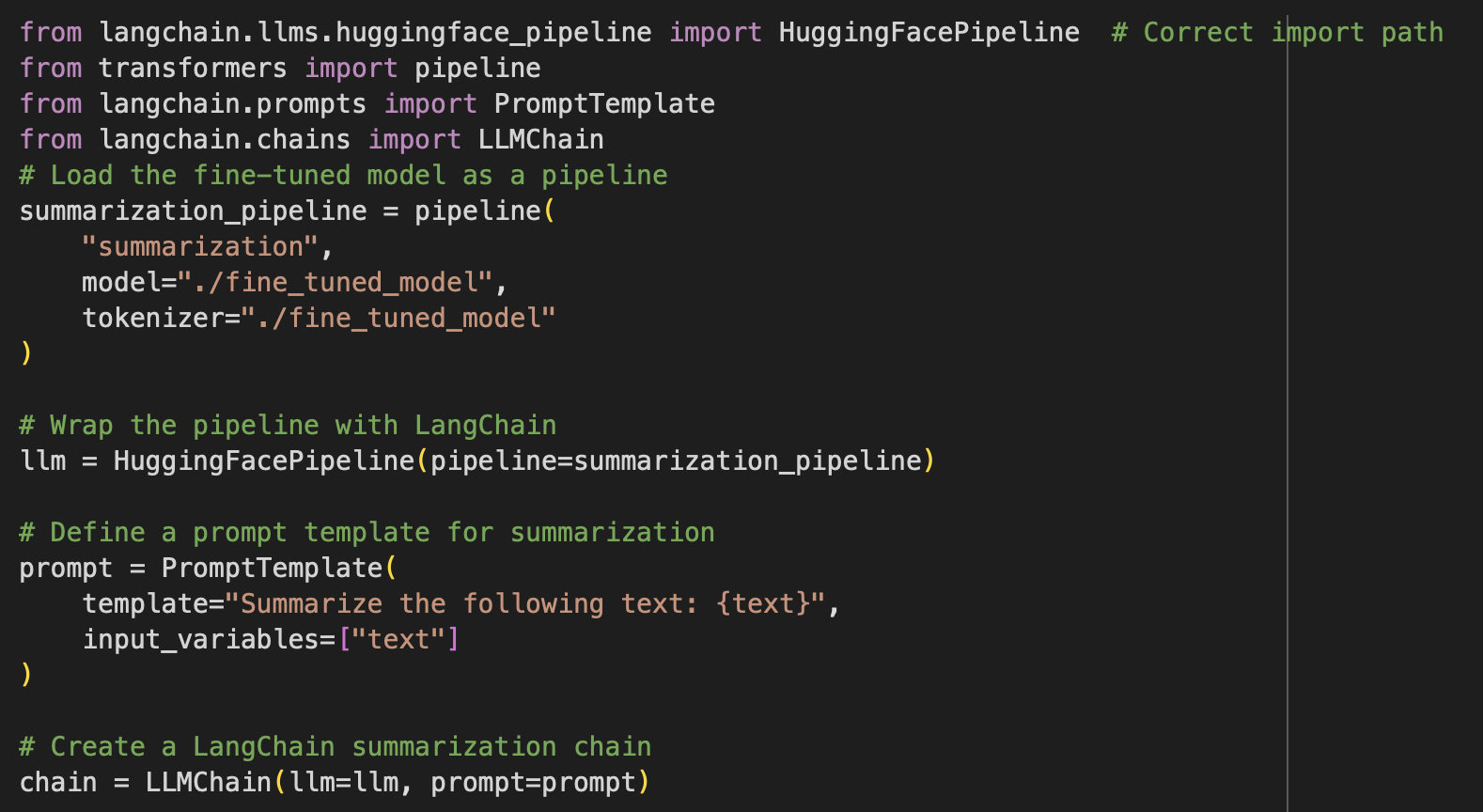
A framework for building applications powered by large language models such as pipelines for summarization. There are three primary advantages to using this framework. It is helpful in data management when using large data and fairly easyto integrate into other models. The most important advantage is prompt engineering; which allows the model to create prompt inputs that are specific to the text in order to help the model learn context and provide better summarization .In order to obtain proper summarization, models need to be fed context in order to understand key points in a text or dialogue to create summaries that match. Langchain uses Prompt. Template to design prompts that direct the model to focus on specific aspects of each text [9]. It is essentially an easy to use interface being wrapped around a fast engine. However, the limitation of langchain is that it is simply a framework and does not come with its own set of pre-trained models[14]. Essentially it boosts whatever model we wrap it around. The model in this scenario is from the HuggingFace Library called T5 Small which is a compact version of the T5 (Text to Text Transfer Transformer) which is a transformer based language model developed by Google. It is an encoder-decoder transformer with close to sixty million parameters, where the encoder processes the



*Figure 8: Langchain workflow*

workflow input and the decoder generates output. Architecturally, there are six encoder and six decoder layers. It can handle a multitude of natural language process tasks such as translation and is known to be a text summarization model.

*Process :*There were three steps to implementing this model and framework. The first step was to fine tune the T5 model. After the initial run, parameters were changed such as learning rate, batch size and sequence length [14]. Then we use Langchain to wrap around the model. This was done as

*Figure 9: Langchain wrapping method* 

shown by the figure above. The last step is to automate text summarization for test data and save the result.

*Hyperparameter Tuning (not all) :*

* Learning Rate: 2e-5 —-> 1e-5
* Epoch:1 —--> 3
* Weight Decay: 0.1 —--- > 0.01

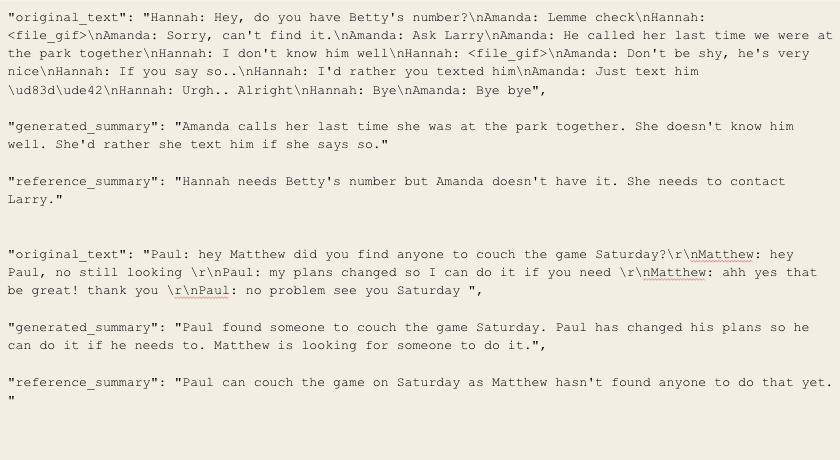
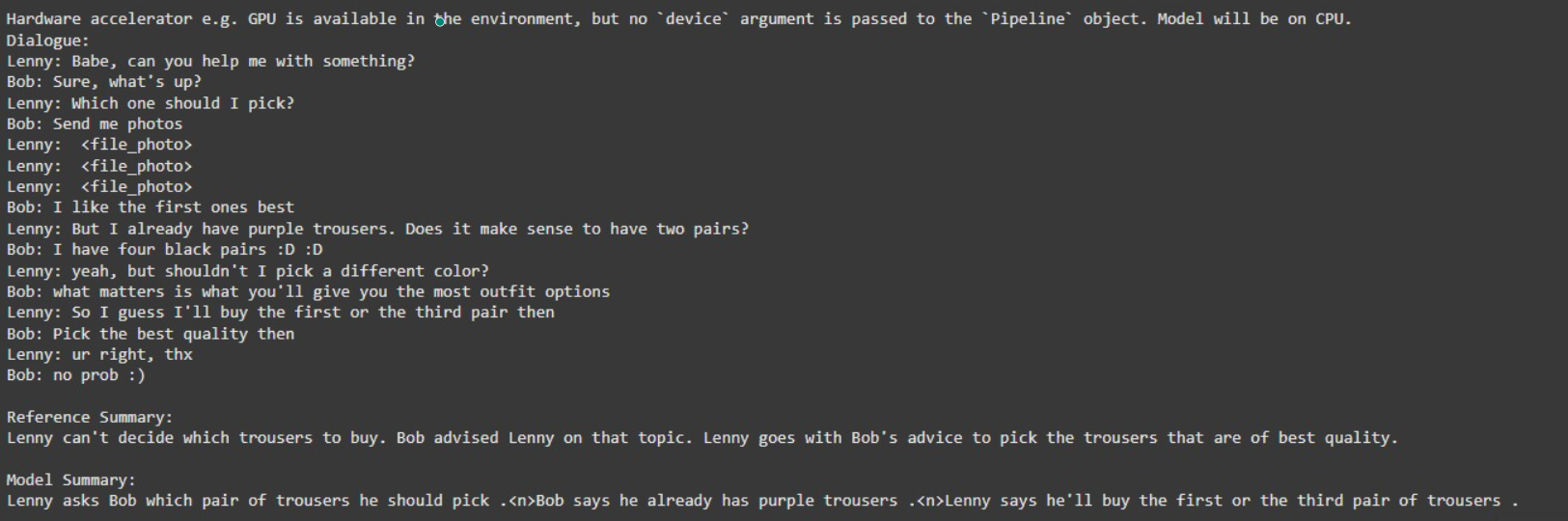
*Challenges:*The primary challenge with LangChain was figuring out how to incorporate the framework, as I had never previously used a wrapping method or integrated a framework with a model[11]. Once the model was fine tuned, learning how to integrate langchain was tough but also was a fulfilling experience as I developed a new skill. We also noticed that the summaries tend to focus on one subject matter which was perfect in some dialogues, however, in other instances where there were multiple topics, the model only summarized the first key topic.

| Dialogue | Reference Summary | Model Summary |
| --- | --- | --- |
| Beatrice: I am in town, shopping. They have nice scarfs in the shop next to the church. Do you want one?\r\nLeo: No, thanks\r\nBeatrice: But you don't have a scarf.\r\nLeo: Because I don't need it.\r\nBeatrice: Last winter you had a cold all the time. A scarf could help.\r\nLeo: I don't like them.\r\nBeatrice: Actually, I don't care. You will get a scarf.\r\nLeo: How understanding of you!\r\nBeatrice: You were complaining the whole winter that you're going to die. I've had enough.\r\nLeo: Eh. | Beatrice and Leo are shopping in town. They have nice scarfs in the shop next to the church. Leo doesn't want a scarf. | *Beatrice and Leo are shopping in town. They have nice scarfs in the shop next to the church. Leo doesn't want a scarf.* |

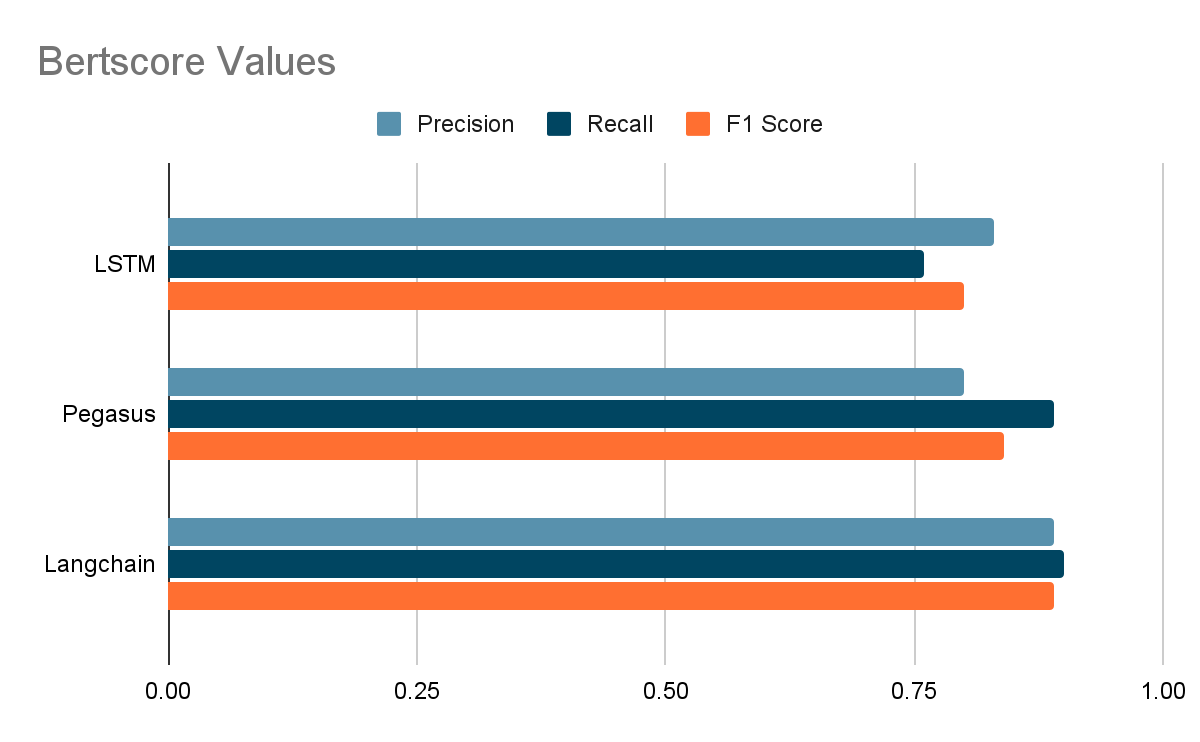
1. **RESULTS**

**BERTScore**

In order to evaluate the results of each model, we employed BERTscore which compares the quality of the generated text to the original text by measuring how similar the two texts are[13]. The similarity between two sentences is calculated by summing the cosine similarities of their token embeddings, enabling the detection of paraphrases. Below are physical representations of the outputs created by the various models we tested. The first two shows generated summaries with the original dialogue and reference summary. The third figure compares metrics of each model using BertScore and the last figure is a table with exact values of precision, recall and F1 Score. We look at precision for quality of positive output , recall for quantity of positive output and F1which combines both.

****

*Figure 10: Pegasus output*   *Figure 11: Langchain with T5 Small output*



*Figure 12:Bertscore Evaluation*

| Model | Precision | Recall | F1 Score |
| --- | --- | --- | --- |
| BiLSTM | 0.83 | 0.76 | 0.80 |
| Pegasus | 0.80 | 0.89 | 0.84 |
| T5 with Langchain | 0.89 | 0.90 | 0.89 |

1. **DISCUSSION**
   1. Each model had its own benefits and disadvantages in complexities, learning curve as well as computational ability. BiLSTM took the longest time to train and was computationally expensive. Langchain boosted the performance of the T5 small model and also had a lower train time. Pegasus provides more abstractive summarizations compared to the other two models.
   2. There are improvements that could be made across all three models. For langchan and T5, incorporating ConversationSummaryMemory, a library from langchain could potentially solve the issue that we face in terms of only picking up on one topic in a multi topic conversation. For BiLSTM, to improve efficiency and solve computational cost, hypertuning the parameters using gridsearch. For Pegasus, continue to tune hyperparameters as we saw an increase in accuracy with changing batch size and optimizers.
   3. Overall**,** all three models had excellent generated summaries. On average, the models were in the eighties range for all three metrics. In terms of precision, all three were above eighty percent accurate which means out of eight hundred and eighteen instances, on average the models generated six hundred and fifty summaries of relevant quality. However, it should be noted that BERTscore is more biased toward models that are alike to itself and also does not account as much for context [13].
   4. ***Presentation Question:***

*Why did the langchain perform better than Pegasus:*

The evaluation metric used in Bertscore which could be a possible reason for “poor” results in Pegasus. Evaluation might perform poorly on tasks that require understanding the context beyond the individual words, such as idiomatic expressions or cultural references. Pegasus tends to provide more abstractive summaries which means it does not focus on using exact words from the input message which could cause the BERTscore to score it lower. The use of specific prompts could have helped create summaries with context and exact words from the reference summary.

1. **CONCLUSION**

The goal of this project was to compare three models on their ability to perform text summarization. We looked at recurrent neural networks through Bi Long Short Term memory model, a transformer based pre-trained model called Pegasus and a transformer model with Langchain integrated as a framework. Each model brought a unique method to accomplish text summarization.While BiLSTM runs both directions in order to obtain past and future context, Pegasus provides summaries based on masking sentences and Langchain provides a framework for the large language models to perform complex tasks such as summarization. Ultimately, we found that T5 Small with Langchain performed well due to the boost provided by Langchain’s prompt engineering which enabled the summaries to have similar context to the original dialogue and summary provided by the dataset.

1. **REFERENCES**

[1] S. Gliwa, P. Mochol, B. Biesek, and A. Wawer, "SAMSum Corpus," Hugging Face, Accessed: Dec. 02, 2024. [Online]. Available: <https://huggingface.co/datasets/Samsung/samsum>

[2] I. Malashin, V. Tynchenko, A. Gantimurov, V. Nelyub, and A. Borodulin, "Applications of Long Short-Term Memory (LSTM) Networks in Polymeric Sciences: A Review," *Polymers*, vol. 16, no. 18, p. 2607, 2024. [Online]. Available:<https://doi.org/10.3390/polym16182607>

[3] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," Neural Computation, vol. 12, no. 10, pp. 2451–2471, 2000.

[4] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in Proceedings of the 3rd International Conference on Learning Representations (ICLR), May 2015. [Online]. Available:<https://arxiv.org/abs/1412.6980>

[5]<https://www.ksolves.com/blog/artificial-intelligence/power-of-langchain-features-and-benefits>

[6] X. Zhang, J. Zhao, and Y. LeCun, "Text Understanding from Scratch," in Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI), Buenos Aires, Argentina, 2015, pp. 1233–1239.

[7] J. Zhang et al., "PEGASUS: Pre-training Transformers for Text Summarization," Google AI Blog, 2020. [Online]. Available:<https://research.google>.

[8] K. Tembhare, "Text Summarization using Streamlit," GitHub repository, 2023. [Online]. Available:<https://github.com/Kunaltembhare003/Text-Summarization-using-streamlit>.

[9] GitHub, "Text-Summarization-Langchain," 2023. [Online] Available:<https://github.com/vmonney/Text-Summarization-Langchain>.

[10] FutureSmart AI, "Summarizing Documents Made Easy With LangChain Summarizer," *FutureSmart AI Blog*, 2023. [Online]. Available:<https://blog.futuresmart.ai>

[11] HackerNoon, "A Comprehensive Guide to LangChain: Building Powerful Applications with Large Language Models," *HackerNoon*, 2023. [Online]. Available:<https://hackernoon.com>.

[12]Turbolab Technologies, "Types of Text Summarization: Extractive and Abstractive Summarization Basics," Turbolab Technologies,Sep.11,2023.[Online].Available:<https://turbolab.in/types-of-text-summarization-extractive-and-abstractive-summarization-basics/>.

[13] Hugging Face, "BERTScore: Evaluating Text Generation with BERT," *Hugging Face Spaces*, 2023. [Online]. Available:<https://huggingface.co/papers/1904.09675>

[14]Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1-67, 2020. [Online]. Available: <https://huggingface.co/google-t5/t5-small>

[15] B. L. S. Varshini, R. Mahadevan, and R. CSP Raman, "Comparative Study and Framework for Automated Summariser Evaluation: LangChain and Hybrid Algorithms," *arXiv preprint arXiv:2310.02759*, 2023. [Online]. Available:<https://arxiv.org/abs/2310.02759>.

[16] A. Retkowski, "A Comparative Study of Transformer Based Pretrained AI Models for Content Summarization," *IEEE Xplore*, 2023. [Online]. Available:<https://ieeexplore.ieee.org/document/10366411>.