Clinical Note Summarization Using T5 and BART

Gaythri Mol Shajimon
Dept. CSEE)
University Of Essex
gi22846@essex.ac.uk

Sifat-E Jahan
Dept. CSEE)
University Of Essex
sj22946@essex.ac.uk

Samuel D Babalola
Dept. CSEE
University Of Essex
sb22912@essex.ac.uk

Suresh R Dodla
Dept. CSEE
University Of Essex
sd22218@essex.ac.uk

Joseph Ejoh
Dept. CSEE
University Of Essex
je19519@essex.ac.uk

Zhen-De-Chen
Dept. CSEE
University Of Essex
sd22218@essex.ac.uk

Sai B Basa
Dept. CSEE
University Of Essex
sb22162@essex.ac.uk

Sylvin J K Gopay
Dept. CSEE
University Of Essex
sd22218@essex.ac.uk

Zecheng Zhang
Dept. CSEE
University Of Essex
sd22218@essex.ac.uk

Vinith Arepally
Dept. CSEE
University Of Essex
va22457@essex.ac.uk

***Abstract*—Utilizing NLP to summarize patients' significant issues from daily progress notes can optimize information overload in hospital management and assist contributors with computerized diagnostic decision support. Health records require a system model to understand, outline, and summarize problem lists. This group project proposes an NLP task for generating a list of problems in a patient's medicare based on the contributor's progress reports during rehabilitation by exploring the performance of the two state-of-the-art seq2seq transformer architectures, T5 and BART. The corpus used here is derived from publicly available progress notes in the Medical Information Mart for Intensive Care (MIMIC)-III. From the result, we concluded that T5 gives a better ROUGE-L score while BART gives a better Bert Score.**

***Index Terms*—BART, T5, Sequence-to-Sequence, NLP, ROUGE-L, Bert Score**

I. Introduction

A patient's electronic health record (EHR) comprises important administrative and clinical data applicable to an individual patient's care. It is a digital form of medical records conserved by healthcare providers over time. The medic's progress reports can be modernized by automating the information access process of the EHR. Other medicare-related activities, including experimental decision-making, quality ensuring, and findings reporting, can be acquired through several terminals. These contain subjective and objective information, are updated regularly, and serve as the most viewed clinical documents. When a patient's illness gets worse, it increases the complexity of the EHR document. Data overcharge recurrently happens in the ICU, with more missed diagnoses and medical errors. Healthcare providers may get assistance to control rational decisions by automatically producing a set of diagnoses in progress notes, to understand a patient's condition through evidence-based medicine more accurately.

It is more difficult to diagnose a patient's problem from daily care notes in ICUs and critical care units. The automatic summarization of daily care notes from the EHR will aid physicians in diagnosing diseases and prescribing medications.

The following are the expected outcomes:

1) To generate a list of problems and diagnoses from the progress notes given by the providers during hospitalization.
2) This diagnostic decision support system is expected to be helpful to clinicians in reducing errors in their diagnosis and improving the efficiency of hospital care.

Natural language processing (NLP) has been demonstrated to be useful for summarising clinical notes in earlier research. In a hospital in New York, HARVEST, an EHR compiler, has been enlarged at the point of care. In the HARVEST NLP model, diseases that are explicitly stated in clinical notes are described by a Markov chain named entity tagger, and the relevance of the mentions is measured by a TF-IDF scorer. Recent work has concentrated on radiology report summaries and doctor-patient conversation summarization with transformer topologies because of breakthroughs in neural approaches. Several studies use transformers to analyze Problem Summarization progress notes to determine and produce the most common diagnosis made while a patient was hospitalized. [1] [2]

The main objective of this task is to summarize the Clinical note using two state-of-the-art sequence-to-sequence models, T5 and BART. Data cleaning and model-specific preprocessing have been done to optimise the system model implementation. To test model accuracy using ROGUE-L and BERT score implementation evaluation matrices.

This document is organised as follows: In section **II**, a literature study on several motivating works is stated in this chapter. section **III** presents a system design including a system overview and different system models of the working procedure of the entire system. In section **IV** Implementation is shown. In section **V**, the testing part is discussed. In section **VIII**, a brief conclusion is stated.

II. Literature Review

While studying BioNLP, T5, and BART and how their mechanism work, many conference papers, journals, and project papers have been analysed. Data gathered from those documents are briefly discussed below.

Abstractive Text Summarization Using BART [3]- In order to determine if this approach would be capable of condensing the massive amount of data, the authors used a BART Deep Learning Model. It contrasts the BART, BERT, T5, and Roberta. The HMSumm structure, which blends extractive and abstractive synopsis, was created by them. The two texts differ from one another and offer diverse meanings in the ROUGE score. Bleu has a score of 0.2 and a number of drawbacks to consider before making copies.

An exploratory study of automatic text summarization in biomedical and healthcare domain [4]-The authors attempted to summarize the public health records using methods already in use, such as Medical Subject Headings (MCH) and Unified Medical Language Systems (UMHS). For the massive records dataset, they utilized tools including the T5 tiny transformer, Tools 4 Noob Source, and Text Compactor. They analyzed the various instruments and found that the T5 tiny transformer provided the best summarization accuracy.

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer [5]-As a unified framework to translate text-based language issues into a text-to-text format using techniques like T5 and C4, the authors developed ”Colossal Clean Crawled Corpus,” where they encountered several obstacles. The objectives of ingesting a set of token IDs corresponding to a tokenized passage of text from their unlabeled text data source are all the outcomes.

How to Summarize Text with Google’s T5 [6]-The writer separated effort into three parts: import & initialization, data & tokenization, and summary production. They also offered a brief introduction to automatic text summarization utilizing PyTorch and Hugging Face as the primary frameworks for constructing text summarizers using T5.

LongT5: Efficient Text-To-Text Transformer for Long Sequences [7]-Using six datasets from CNN/Daily Mail, PubMed, arXiV, BigPatent, MediaSum, and Multi-News, the authors created Long T5 to examine the effects of scaling input length and model size. Long T5 and a number of popular techniques, such as BigBird-PEGASUS, HATBART, DANCER PEGASUS, PRIMER, TG-MultiSum, LED, and a BART application, are contrasted in terms of their findings.

Text Summarizers [8]- They gave a comprehensive overview of NLP in the paper ”Text Summarizers,” which is based on cutting-edge Deep Learning and Machine Learning research. Text summarization methods mechanically distil the most important soundbites from texts, papers, podcasts, movies, and other media. It is possible to use Speech-to-Text APIs for both audio and video streams, including podcasts and YouTube videos, as well as for static, pre-existing texts, such as research papers or news stories.

Text Summarization [9]- This article provides a quick explanation of natural language processing, which encompasses both abstract and extractive summarizing. Text ranking, grouping, and feature-based algorithms are among the main techniques used in extraction. Sequence-to-Sequence Models and Pointer-Generator Networks are the techniques employed in Abstraction. The process of text categorization includes processing, text representation, model training, summary, and assessment.

An approach for extractive text summarization using fuzzy evolutionary and clustering algorithms [10]- The approach to text categorization extraction presented in this paper, The authors want to develop an extractives-based summarization model that can be used with both individual texts and groups of documents.

Deep contextualized embeddings for quantifying the informative content in biomedical text summarization [11]- The purpose of this study, is to examine the efficacy of a state-of-the-art deep contextualized language model for summarizing biomedical text materials.

Leveraging Pretrained Models for Automatic Summarization of Doctor-patient conversations [12]-The authors investigate the viability of automatically summarising doctor-patient conversations using transcripts utilizing pre-trained transformer models. The best human reference typically surpasses generated summaries in missing and hallucination, with the missing score being the lowest of all quality criteria, illustrative of the frequent false negative errors made by the carefully designed algorithms.

MultiGBS: A multi-layer graph approach to biomedical summarization [13]- The researcher behind this study claims that. Current text summary techniques sometimes exclude important details by choosing sentences based on just one textual element. The unsupervised method uses the Multi-Rank algorithm and the number of ideas to choose words from the multi-layer network. Last but not least, the suggested approach scores the sentences in the input document using the Multi-Rank algorithm on a multi-layer network. The suggested solution manages a variety of linkages between the text’s words by using multi-layer graphs rather than basic graphs. It is a cutting-edge biomedical summarization device.

Summarization & Generalization of Discharge summary medical reports [14]- In this essay, the author makes the suggestion that a discharge report be created using the nursing notes that were recorded throughout the patient’s hospital treatment. By condensing nursing notes and using them to construct parts of the discharge summary report, they would like to extend these goals to healthcare professionals, who will then be able to quickly study the patient’s file and make prompt medical decisions.

Bio BART: Pretraining and Evaluation of A Biomedical Generative Language Model [14]- The goal of this study is to put out a ground-breaking pre-trained language model that can enhance the production of excellent biomedical writing while also advancing the field. The authors succeeded in creating a pre-trained language model for the biomedical domain that can produce high-quality text for a variety of applications by combining data pre-processing, architectural design, pretraining, fine-tuning, and assessment procedures.

Abstractive Summarization of Radiology Reports using simple BART Fine Tuning [15]- They participated in the summary of the radiology report by sharing experiments and findings. They modified BART on the training set before testing it on the development set. Overall, using the BART base as the pre-trained model yields the best test set results. In this post, we discuss all of their experiments for fine-tuning pre-trained models for summarising radiological results.

Out of the Box” Information Extraction: a Case Study using Bio-Medical Texts [16]- Due to the difficulty of scientific language compared to that of standard Web text and the low redundancy of our bio-medical texts, they anticipated ReVerb to provide only modest accuracy in this work. In their experimental case study, they used passages from the biology textbook used for the HALO project and papers from MedLine to test the ReVerb system. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension* [18]- The study’s findings show that the BART model, with its cutting-edge performance on a variety of benchmarks, is a very effective solution for a wide range of natural language processing applications. These results have led researchers and practitioners in natural language processing to favour the BART model.

III. System Design

The main objective of our project was to implement a model which is capable of summarizing the important diagnosis mentioned in the progress notes provided by the carers, nurses, or physicians. We have used two state-of-the-art sequence-to-sequence models T5 and BART for the implementation. After the summarization of progress notes the entire predictions will be exported to a CSV file.

A. System Architecture

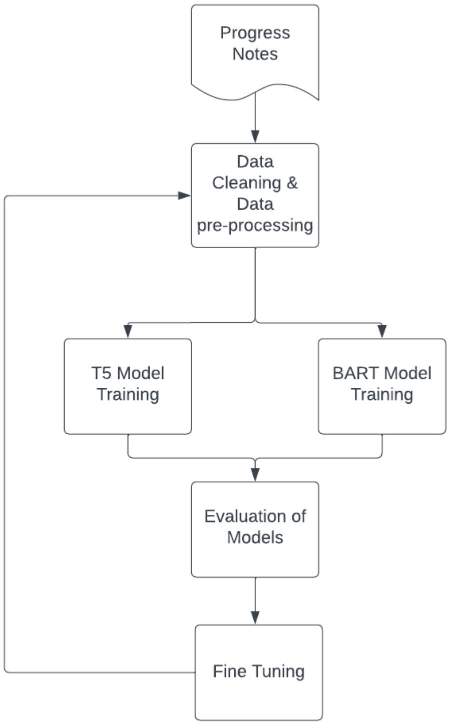

Fig. 1: T5 and BART end-to-end architecture

The following are the main components of the system architecture(Figure.1):

- ***Progress Notes***: The architecture starts with an input CSV file, which contains the progress notes of patients admitted in different sections of the hospital. It is structured in SOAP format. It is presented below
  a) *Subjective Section*: It contains the health problems expressed by the patient like major complaints along with past medical complaints and also medical history. It also contains the different medical tests done for the patient.
  b) *Objective Section*: It includes vital signs along with the medications given to the patient and laboratory test results.
  c) *Assessment*: It is the main assessment provided by the carers, nurses or physicians. It contains the active and passive diagnosis. The reason for admission will be described here and also contains the active problems faced by the patient.
  d) *Plan (Summary)*: It is the final summary given by the physician, which is the ground truth for training the model. It will be multiple subsections containing treatment plans and medical problems.

  Along with the above data a column called File id will be there in the CSV file, which is very important as it is the unique id corresponding to each patient's file.

- ***Data Cleaning and Preprocessing***: The provided CSV file needs a lot of cleaning process as it contains a lot of special characters, de-identified words, punctuations, stop words etc. So we have done extensive cleaning before passing it to the model. The detailed explanation of different data cleaning and preprocessing we have done is described in section 3.

- ***T5 Model Training***: T5 is one of the pre-trained models we have used for the training. It is developed by Google AI Language. It is capable of performing different types of NLP tasks mainly text generation, summarization, translation and more. It is pre-trained on the massive corpus, where it is trained to generate a wide range of text output by predicting masked words in the input sentences. T5 can be fin-tuned with a relatively small amount of task-specific training data. Moreover, it is a powerful and highly flexible language, which achieved state-of-the-art performance in a wide range of NLP tasks. The implementation of the T5 model will be explained in section 3.

- ***BART Model Training*** : BART model is developed by Facebook AI. It is a state-of-art sequence-to-sequence model which is capable of performing varieties of text generation tasks including machine translation, text summarization and text generation. It used a self-attention mechanism to find the dependencies in a different part of the text. It is also trained using masked language modelling objectives and denoising auto encoding, it helps the model to learn rich representations of input text, which can be used for different text generation tasks.

- ***Evaluation***: The model was evaluated using the ROUGE-L score and Bert Score. ROUGE-L is a metric used for the evaluation of the longest common subsequence on a generated summary or machine translation system. n-gram overlapping is used in the comparison. It will give us a measure of the amount of reference summary captured in the predicted summary.
  Bert Score is also another metric used for measuring the quality of predicted summary, in which it will also take semantic meaning into account for the scoring. It will check the cosine similarity between the predicted summary and the original summary and gives a score between 0 and 1.

- ***Fine Tuning*** - The fining tuning of the model has been done by changing the number of batches, changing the learning rate, the number of epochs, changing the maximum length of input token and summary and making changes in the preprocessing steps.

The input file will be provided or uploaded by physicians to the models. Once the file is uploaded data will be cleaned and preprocessed using the steps mentioned in section 3 After preprocessing the input data will be passed through the model and model training will be done. Once the training is completed the final submission file will be generated and saved in the required location. We have two options for input, one is to produce a diagnosis only using the Assessment column and another option is to use Assessment and Subjective section in the input CSV file.

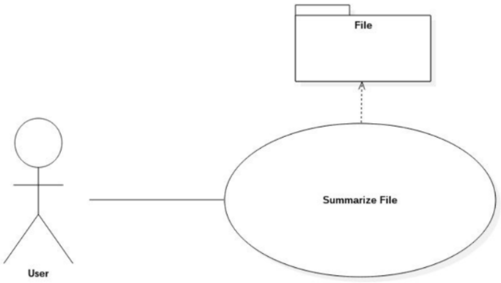

Fig. 2: Use Case Diagram

The use case diagram in fig.2 [19] illustrates below:

- The user will upload the file
- The summarization algorithm will take the input file (progress notes)
- Perform prepossessing and model training
- File with summarized predictions will be generated

The above workflow has been implemented with 5 files, 2 notebook files for exploratory data analysis and Model Training and results respectively and 3 python files for data preprocessing, dataset and data loader creation the third one is created as a utility class which contains the repeatedly using functions. A details explanation of implementation will be described in the following section.

IV. Implementation

This section contains a detailed explanation of the experiment set-up and code implementation.

A. *Experiment SetUp*

The basic requirement for the implementation and execution of the code is

1. A high-performance machine with 38.7 GPU and 54 GB memory. The coding has been done in the google colab pro account with the runtime as Premium and high memory.

2. To upload the required python files in google drive for the model training.

3. Python is the programming language used with two state-of-art sequence-to-sequence models T5 and BART has been used for the model training. Pytorch machine learning library is used for model training.

B. *Key Components of the system*

a) **Exploratory Data Analysis**: The initial file in the implementation is exploratory data analysis which contains the data visualization of input and labels. This file is important as using which we can decide what all data cleaning and data preprocessing steps need to be taken before training the model. This is a separate notebook file which will run separately. Some of the data visualizations in exploratory data analysis are in Fig.3 and Fig.4. By comparing the figures, it will be easy to understand the most frequent words in the text.

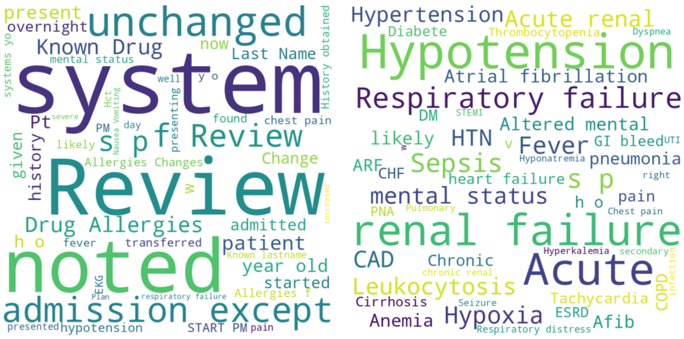

Fig. 3: word cloud image of input and summary

b) **Data Pre-processing class**: Data cleaning and preprocessing is the most important step that needs to be performed before the training of the model. Different models require specific preprocessing steps to be done before training. For example, the T5 model requires adding the prefix ’summarize’ while doing summarization tasks. T5 is capable of training the model with less preprocessing compared to BART. For BART, the accuracy of the model increases with more fine-tuning. Some of the preprocessing steps performed are described below:

a) *Check for missing and duplicate values*: The data contains some missing values in the summary or assessment or subjective section columns, and this was dropped in order not to affect the evaluation.

b) Consecutive duplicate words and extra spaces - All the extra spaces were replaced with one space and consecutive duplicated words which makes incorrect meaning has been removed.

c) Removing personal identifiers: The input data contains a lot of personal de-identifiers, as it contains the patient’s details. Before the data is used for analysis, any identifiers that may jeopardize the patient’s privacy must be eliminated. Personal identifiers removed from the data include patient names, social security numbers, and phone numbers. In our clinical notes data, identities like Mr., Mrs, Dr etc. were also removed.

d) *Removing stop words*: All human language has an abundance of stop words. By getting rid of these words, we can make our text more focused on the key information by eliminating the low-level information. Because there are fewer tokens involved in training, the removal of stop words obviously reduces the size of the dataset and, consequently, the training time. Both the general stop words in the library and additional stop words (defined by us) were removed. Additional stop words which are not contributing value t results were also removed.

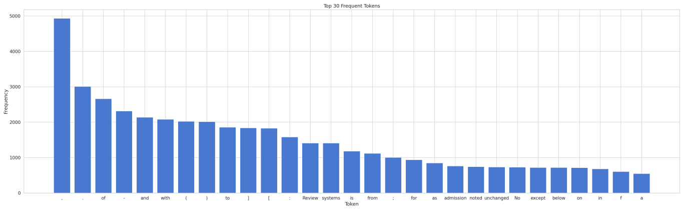

Fig. 4: Word-frequency count of input

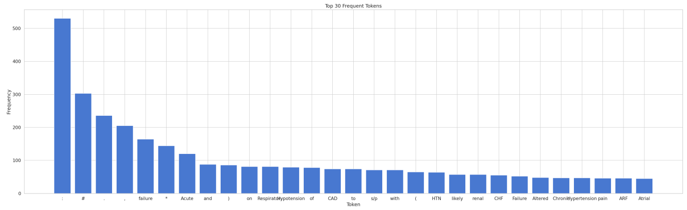

Fig. 5: Word-frequency count of summary

e) *Normalization of Data* - Data normalization entails putting the data into a consistent format so that it may be easily studied. Data normalization for clinical notes data may involve deleting dates if it were only assumed, numbers with special characters, age and gender representation and other elements that can make the data challenging to evaluate. There are differences in the date format utilized in the clinical notes data, and this was cleaned by replacing all the date formats in the clinical notes with the placeholder 'date'. The age and gender representation in the data were replaced with the placeholder 'male' and 'female' for both genders respectively. Additionally, we encountered some varying de-identified words in the texts. These varying words were defined, and all were removed. The data became consistent and simpler to evaluate when the dates and de-identified words were normalized to a standard format.

f) Dropping all columns except File id, Assessment, Summary and Assessment + Subjective Section: A new column with concatenated data of Assessment and Subjective section has been created. After that Subjective section and objective section of the data frame have been removed as it is no more used.

c) **Data Set and Data Loader Creation** - Before training the model, the data frame needs to be converted into a training, validation and test dataset and needs to convert it to train, validate and test data loaders.
Using a set of rules, a tokenizer divides the text into tokens. Tokens will be transformed into numbers, which will then be transformed again into tensors, and this will serve as the model's input. Any required additional input will be added by the tokenize. It gives back a dictionary with these 3 things: a. *input ids* b. *Attention mask* and c. *Token type ids*.
Two datasets were created, one for training and one for validation. 80% of the dataset is the training dataset, which is used to fine-tune the model. The validation dataset will be used to assess the performance of the model.

d) **Model Training** - There are five model with 2 different input(Assessment Only and Assessment + Subjective section) was used for training. The five models used for the training and training details are as follows:

   a) T5-small - It is the smallest version in the T5 family with a number of parameters equal to 60 million. Despite its size, it very powerful model that can be fine-tuned for various NLP tasks.

   b) T5-base - It is a mid-sized T5 model, where it contains 220 million parameters and is more powerful than T5 small and this can be also fine-tuned for various NLP tasks like text summarization.

   c) T5-Large - This is the largest pre-trained model in the T5 family. It has 770 million parameters. It is used to perform more complex tasks NLP tasks.

   d) BART-base - BART base is the smallest in the BART family with a number of parameters equal to 140 million and it is based on transformer architecture. BART can be also fine-tuned for different NLP tasks.

   e) BART-Large - This is the largest pre-trained model in the BART family with 400 million parameters. It is more suitable for complex NLP tasks such as long-form text generation, advanced dialogue systems and summarization tasks.

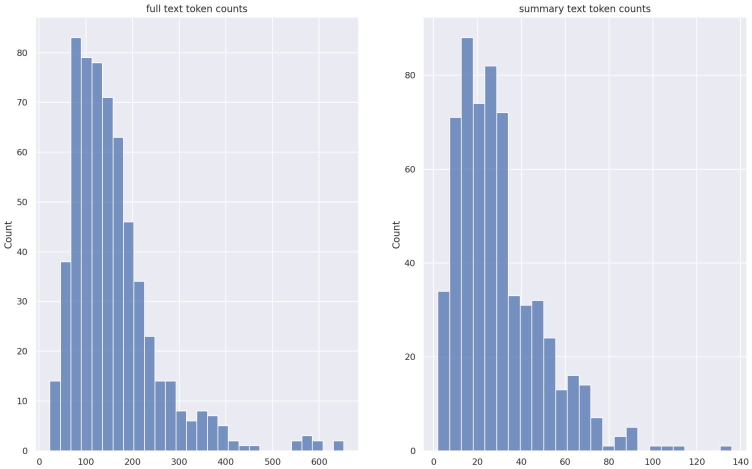

Fig. 6: Token count after T5-base tokenization of input and summary

Model training starts with input from the data loader. The model needs to be configured according to its size of the model. Some model such as T5-small needs more epoch compared to T5-base. We need to configure the epoch according to the model that has been trained. Batch size is configurable, where we can configure it to a batch size of 8, 16, 32 etc. After configuring the epoch and batch size the corresponding tokenizer of the model will tokenize the provided data frame and pass training, validation and test data frame along with the tokenized data to the data loader class. The data loader class will convert it to the required format for the respective models. The input will be passed to the model with the model setup created using the model class to the trainer.

Trainer is configured to store the best checkpoints. Early stopping criteria have been defined in such a way that if two consecutive validation losses higher than the lowest validation happened, then the training will be stopped. The tensor board has been initialized and used for logging the training and validation loss.

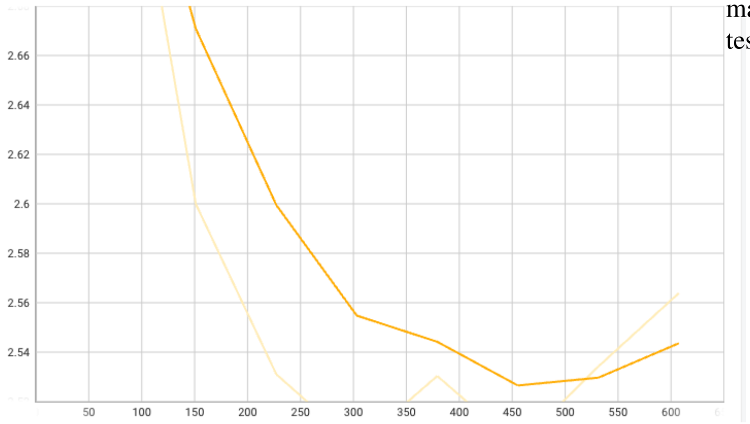

Fig. 7: T5-base Validation loss

e) **Model Evaluation** :
After the training of the model, the model will be evaluated using the evaluation matrices using ROUGE-L and Bert Score. And again fined tuned by tweaking the learning rate, batch size, number of epochs, and the maximum length of token in input text and summary. The evaluation results will be described in a separate section below.

Overview of code listings :

a) *Exploratory Data Analysis.ipynb* - This file contains the data visualization and exploratory data analysis of input and summary.

b) *datapreprocessing.py* - This file contains the class which is reusable for all models for performing data cleaning and preprocessing tasks. The preprocessing steps performed in this class are explained in the above section.

c) *clinical_Note_module.py* - This file contains 3 classes, the clinical note class returns the dataset and the clinical module contains calls clinical_note class and creates data loader and return. The clinical_note_model class inherits from the PyTorch lightening module. It contains the feed-forward pass of the model and also it contains the training and validation set-up for the model.

d) *utility.py* - This class acts as a utility class, where it contains the function for generating summary, trainer initialization, evaluation, and function for displaying tokens.

e) *Main.ipynb* - This is the main notebook where models will be trained and evaluated.

V. Testing

The testing has been done using the above-described evaluation matrices ROUGE-L and Bert Score. Following are the different test cases we have done for hyperparameter tuning.

a) Learning Rate - The learning rate which gives the best accuracy is .0001, tried with .001 and .00001, but accuracy was decreasing. So configured the learning rate with .0001.

b) Batch Size - Tried with different batch sizes 8, 16 and 32, but 8 gives a more accurate result.

c) Maximum_length_token - The maximum length of input and output has been tried with different numbers 512, 700, 600 etc. 512 gives better results.

d) Epoch - The number of epochs has been changed and tested, finding that smaller models required more epochs than the large models with the same learning rate.

e) changing preprocessing steps - By trying out different preprocessing steps tried to train the model. The analysis shows the accuracy changes according to the changes in preprocessing methods.

After doing all the evaluation steps, it is found that T5-Large with assessment and subjective section is giving a better ROUGE-L score while BART-base with Assessment gives a better Bert-Score. The table below shows the test

Table 1. ROUGE-L Score & Bert Score

Models	**ROUGE-L**	**Precision**	**Recall**	**F1-Score**
T5-small A	0.140312	0.600255	0.491515	0.534391
T5-small A + S	0.117785	0.586044	0.490910	0.528657
T5-base A	0.106157	0.574313	0.481526	0.516567
T5-base A + S	0.1573363	0.614515	0.527568	0.516567
T5-large A	**0.159587**	0.596772	0.475150	0.521213
T5-large A + S	0.150353	0.596061	0.487464	0.528889
BART-base A	0.100197	0.618254	0.584031	0.597105
BART-base A + S	0.108703	0.626793	0.589254	**0.602401**
BART-large A	0.020438	0.456955	0.524455	0.487583
BART-large A + S	0.066984	0.509786	0.559990	0.532254

We got a maximum Bert F1 score for the BART-large model with a score of 0.599719 and a maximum ROUGE-L Score of .1585 for T5 large with the Assessment and subjective section.

VI. Security Aspect of the Project

The security of a product has great relevance nowadays, this needs to be ensured before making it available to the public or end users in order to avoid problems like data breaches, and load issues. Some of the security aspects related to our project are as follows:

- *Data Privacy* - Clinical notes contains may contain sensitive information like medical history, treatments and diagnosis. It is important that we have made sure of the patient's privacy during the summarization process. Including encryption techniques for data protection is one way of solving this issue, and it can be considered as a future scope of this project.

- *Data Breaches* - Proper measures need to be taken to prevent the data breaches, such as anti-virus software, implementing firewalls and other security protocols.

- *Model Bias* - T5 and BART model is prone to bias depending on the training data. In order to avoid this we should use diverse data during the training.

VII. Project Management

Project management plays important role in the successful completion of a project. To a great extent, we have succeeded in managing group activities. The success of a group project depends on how the team handled the problem and solved the issues. Our team consist of 10 members with the different skill set and the majority of us are new to NLP. There is an initial struggle is there in the team to understand the concept, but we helped each other to understand and upskill team members, wherever required. The following section contains a detailed explanation of how we managed the group activities:

- *Splitting of Task*: We have created separate groups for each of the tasks namely, Background Study and Literature review, Pre-processing, Development, Testing and Report Writing. Project management we used includes Gantt chart and Jira. We have regularly created tasks in the Jira board and completed most of the tasks on time. Following is the division of tasks in each sprint in Jira.
  - *Sprint 1*: In this sprint, we created tasks for finding suitable project ideas for the project and we got many innovative ideas from team members we shortlisted 2 of them and presented them to the supervisor.
  - *Sprint 2*: In this sprint, we have done a brainstorming session on the project topic given by the supervisor. For getting the dataset for the project we have to complete certain prerequisites like completing CITI training, getting access to the MIMIC III dataset etc. We completed all those constraints and got the dataset required for the project.
  - *Sprint 3*: After getting the dataset we focussed on the completion of the requirement specification document and we completed it on time and submitted it.
  - *Sprint 4*: We created 4 groups specifically for Literature review and background study, Preprocessing, development and testing. The development team started learning about T5 and BART models, the background study was done by another team, and parallelly preprocessing team worked on their task.
  - *Sprint 5*: The development team started implementing the T5 and BART respectively and the literature review team continue work on their task. Pre-processing team has given the initial deliverable to the development team. The development team used this along with some additional preprocessing for model training.
  - *Sprint 6*: One dedicated person for project management has been allotted and other team members continued working on their respective tasks.
  - *Sprint 7*: T5 implementation got completed and BART implementation was in progress in the development team, testing team, literature review team and preprocessing team merged to the final report writing team.
  - 8, *Sprint 8*: In this sprint, BART implementation got completed and final testing was also completed and presented our result to the supervisor.
  - *Sprint 9*: Everyone worked on the report and initial draft shared with the supervisor and worked on the feedback given by the supervisor.
- *Group Meetings* - We have regular meetings every Tuesday and also one more extra meeting on Thursday, which is only for 30 minutes to clear doubts and remove blockers in the assigned task. During the group meeting, we discussed the status of the assigned task and check if anything is pending. Also, assigned new tasks according to Gantt Chart. Chairperson and Secretary have been selected for each week, the secretary is responsible to write group meeting minutes and the chairperson is responsible for taking decisions and managing the team. We have a biweekly meeting supervisor and we never missed any meetings with him and we followed the feedback given by the supervisor.
- *Group Management* - We have a separate meeting group created for each main task. The person who is responsible for project management is there in all the meetings happening in the subgroups and he checks the progress of the tasks, makes sure that all team members have tasks assigned for the week and also he will create and update the task in Jira. If there are any blockers in the team, he will report it to the chairperson, first, it will be discussed at the team level and if it is not solved, he will communicate it with the supervisor to get it solved.
- *Merging the subtask* - We have done development parallelly, so our work in GitLab is uploaded in a separate feature branch. After the whole coding is completed in exploratory data analysis, preprocessing and model implementation, a separate class for preprocessing has been created and imported into the model training file. Exploratory data analysis has been kept as a separate file. We have created 2 more pythons for other major functionalities like dataset creation, data loading, model setup and summarization tasks. All these files are imported in the main jupyter file where we do model training. All the 3 file files and 2 Jupytor files merged to master in sprint 8.
- *Change Management*: We have done some changes to the initial plans after getting approval from the supervisor. The first change was, initially, we planned to use T5 and Clinical_BERT for the implementation, but we later changed Clinical_BERT to BART as Clinical_BERT is not a sequence-to-sequence model. The second change is assigning project management tasks to a dedicated person, this got started from sprint 5, as per feedback from the supervisor.
- *Risk Management* - In order to reduce the risk of loss of

work we regularly uploaded all our work to git lab and kept a copy in the group google drive. Also in order to manage the work in the group due to a medical emergency or unavoidable circumstances, we always make sure that all the group members are aware of the work done by each one.

VIII. Conclusion

Overall the experiment is a success, we were able to score more Bert Score than the existing system with a Bert Score of nearly .6 in the validation dataset. In the existing system maximum, Bert Score was .5797. The major challenge we faced was the computational charges enquired as model training requires nearly 57 GB memory and 38 GPU for completing all the training and it also took 1.5 hours to complete the whole training. Other than this the experiment was smooth and the result was satisfactory. In future, expected to get much better results if we add additional preprocessing steps.

The methodology and tools we chose for the training were correct and gave good predictions.

The project management was effective and successful. We divided tasks among the team members as literature review, preprocessing, development, testing and report writing. We completed the task as per the milestone mentioned in the Gantt Chart. We have created tasks on the Jira board every week and completed the specific task. If there is any blocker we will discuss it as a team and resolve the issues. Other than the regular meeting we have an additional meeting on Thursday to make sure that there are no blockers in the assigned task. We have a dedicated person only for project management, who is responsible for creating tasks in Jira and coordinating with each team.

References

[1] Gao, Y., Dligach, D., Miller, T.A., Xu, D., Churpek, M.M., & Afshar, M. (2022). Summarizing Patients' Problems from Hospital Progress Notes Using Pre-trained Sequence-to-Sequence Models. Proceedings of COLING. International Conference on Computational Linguistics, 2022, 2979-2991.

[2] Bryant Furlow. 2020. Information overload and unsustainable workloads in the era of elec2988 Electronic health records. The Lancet Respiratory Medicine, 8(3):243–244.

[3] A. Venkataraman, K. Srividya and R. Cristin, "Abstractive Text Summarization Using BART," 2022 IEEE 2nd Mysore Sub Section International Conference (MysuruCon), Mysuru, India, 2022, pp. 1-6, doi: 10.1109/MysuruCon55714.2022.9972639.

[4] Mukesh Kumar Rohil, Varun Magotra, An exploratory study of automatic text summarization in biomedical and healthcare domain, Healthcare Analytics, Volume 2, 2022, 100058, ISSN 2772-4425, https://doi.org/10.1016/j.health.2022.100058.

[5] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. ArXiv. /abs/1910.10683

[6] Medium, How to Summarize Text With Google's T5. https://betterprogramming.pub/how-to-summarize-text-with-googles-t5-4dd1ae6238b6

[7] Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. LongT5: Efficient Text-To-Text Transformer for Long Sequences. In Findings of the Association for Computational Linguistics: NAACL 2022, pages 724–736, Seattle, United States. Association for Computational Linguistics.

[8] Medium, The Power of Text Summarization Streamlining Insights. https://medium.com/mlearning-ai/the-power-of-text-summarization-streamlining-insights-c2be01e3a9f0

[9] Medium, The Power of Text Summarization Streamlining Insights. https://medium.com/mlearning-ai/the-power-of-text-summarization-streamlining-insights-c2be01e3a9f0

[10] Pradeepika Verma, Anshul Verma, Sukomal Pal, An approach for extractive text summarization using fuzzy evolutionary and clustering algorithms, Applied Soft Computing, Volume 120, 2022, 108670, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2022.108670.

[11] Moradi, Milad Dorffner, Georg & Samwald, Matthias. (2019). Deep contextualized embeddings for quantifying the informative content in biomedical text summarization. Computer Methods and Programs in Biomedicine. 184. 105117. 10.1016/j.cmpb.2019.105117.

[12] Longxiang Zhang, Renato Negrinho, Arindam Ghosh, Vasudevan Jagannathan, Hamid Reza Hassanzadeh, Thomas Schaaf, and Matthew R. Gormley. 2021. Leveraging Pretrained Models for Automatic Summarization of Doctor-Patient Conversations. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 3693–3712, Punta Cana, Dominican Republic. Association for Computational Linguistics.

[13] Ensieh Davoodijam, Nasser Ghadiri, Maryam Lotfi Shahreza, Fabio Rinaldi, MultiGBS: A multi-layer graph approach to biomedical summarization, Journal of Biomedical Informatics, Volume 116, 2021, 103706, ISSN 1532-0464, https://doi.org/10.1016/j.jbi.2021.103706.

[14] Pal, Koyena. "Summarization and Generation of Discharge Summary Medical Reports." (2022).

[15] Hongyi Yuan and Zheng Yuan and Ruyi Gan and Jiaxing Zhang and Yutao Xie and Sheng Yu, BioBART: Pretraining and Evaluation of A Biomedical Generative Language Model, arXiv:2204.03905, 2022, https://doi.org/10.48550/arXiv.2204.03905

[16] Ravi Kondadadi, Sahil Manchanda, Jason Ngo, and Ronan McCormack. 2021. Optum at MEDIQA 2021: Abstractive Summarization of Radiology Reports using simple BART Finetuning. In Proceedings of the 20th Workshop on Biomedical Language Processing, pages 280–284, Online. Association for Computational Linguistics.

[17] Balasubramanian, N., Soderland, S., Mausam, Etzioni, O., & Bart, R. (2013). " Out of the Box " Information Extraction: a Case Study using Bio-Medical Texts.

[18] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. Annual Meeting of the Association for Computational Linguistics.

[19] Team 13. Software Requirement Specification Document by Team 13

Appendix

a) User Document - Access Link
b) Installation Instructions - Access Link
c) Project Code - Access Link
d) Group Meeting Minutes - Access Link

Note: We have all four above items in faser as well