Lay Summarization for Biomedical articles using Transformers

[1]Shivam Rajput (2022H1030058H)

[2]Prakhar Yadav (2022H1030081H)

[3]Yogesh Singh Thakur (2022H1030073H)

[1,2,3] Department of Computer Science & Information System, BITS Pilani Hyderabad Campus, India.

I. Abstract:

Biomedical publications cover the latest research on health-related topics that can be of interest to various groups such as researchers, medical professionals, journalists, and the public. However, these articles are often long and technical, making it challenging for a vast audience to understand them. To overcome this issue, Lay Summarization is necessary to summarize the main points of the article using simpler language. Abstractive summarization models can be used to generate a brief summary that contains more background information and less technical terms than the original text. To further enhance the results, our research work developed various architectures for Lay Summarization of medical articles. In this report, we explain the data cleaning and preprocessing steps taken and the model architecture.

II. Introduction:

Scientific publications play a crucial role in advancing our understanding of various scientific disciplines. However, these articles can be challenging to understand as they often contain specialized language and require prior knowledge. This can prevent readers who lack expertise from fully comprehending the research and limit the impact of the findings to a small community. Moreover, it can lead to misunderstandings among the public and journalists, which can have adverse effects. Therefore, it is essential to communicate scientific findings in a manner that is accessible to a broader audience.

To address this issue, several initiatives have been taken to make scientific publications more accessible. Some journals now require authors to provide a summary of their research in layman's terms. Additionally, there are now several online platforms that offer plain-language summaries of scientific articles. Furthermore, recent advancements in natural language processing and machine learning have facilitated the development of automated tools for summarizing complex scientific articles into simple, easy-to-understand language.

However, despite these efforts, there is still a need for more effective communication of scientific findings to the general public. As such, researchers and publishers should continue to explore innovative approaches to make scientific publications more accessible to everyone.

The significance of communicating scientific findings in a manner that is accessible to the public is even more critical in the case of biomedical research. Biomedical research involves highly technical and constantly evolving terminology that can be challenging to understand for the layperson. Furthermore, the findings of biomedical research can directly impact public health, making it imperative to ensure that the public has access to accurate information. The COVID-19 pandemic is an example of how misinformation and misunderstandings can have catastrophic consequences.

To address these challenges, some academic journals have taken the initiative to publish lay summaries that explain the context and significance of a research article in simple language that can be understood by the general public. These summaries are designed to complement the technical content of the article and make it accessible to a broader audience. Additionally, recent advancements in natural language processing and machine learning have facilitated the development of automated tools for generating lay summaries of biomedical

research articles. These tools are capable of summarizing technical articles into a concise and readable format that can be understood by a wider audience.

Automated summarization techniques can be useful in creating lay summaries for scientific articles. While summarization techniques have been applied to scientific articles before, most of these efforts focused on generating technical summaries. There are only a few studies that have dealt with the task of generating lay summaries, and only a limited number of datasets have been created for this purpose. Additionally, the datasets that do exist tend to use specific article and summary formats that may not be applicable to other biomedical literature. These issues slow down progress in the field and hinder the development of models that can make scientific content more accessible to a wider audience.

A lot of these problems were solved by the work of Goldsack et al when they produced a gold standard dataset specifically for this task. We shall discuss more about it in the Dataset section. Based on this dataset, various architectures were developed. Initial approaches consisted of Lead K based basic models. The best results were provided by supervised BART models achieving a ROGUE 1 score of 0.42 on the PLOS dataset.

Our aim with this project is to fine tune the models and come up with an improved architecture to produce better lay summaries. We will be using the dataset provided by the BioLaySumm shared task. We designed and tested various different approaches and found varying albeit mediocre results in the process.Our highlight being developing a low data model which may help generate Lay summaries of other fields which do not have large datasets.

III. Literature Survey

We looked into related work particularly for the medical articles, which are listed below:

A. Approaches for lay summarization in Medical field

There have been comparatively less work then other fields and results are not very good. Lay summarization for medical domain require large corpus of articles and reports strictly verified by professionals. After the success of general text summarization Neural networks and Pre-trained models(PLMs) have been explored in recent years. Cohan et. Al 2018[2], Wang at el 2021[3] summarizes the current trends of Biomedical filed of summarization. As per [2] BERT and BART giving the promising results. The attention-based encoder-decoder network has been improved by many. The attention masking of the Long-Short Term Memory (LSTM) with two different networks: pointer-generator network that produces accurate expression by pointing each word in the source and coverage network that avoids repetition. Liu and Lapata 2019 [4] used BERT model as an encoder, Zhang et al. 2019a [5] applied BERT to both encoder and decoder networks, Scialom et al. 2020 [6] constructed generative adversarial networks using BERT models, and Yoon et al. (2020) appended semantic similarity layers on top of the pre-trained BART. In [9] authors truncated all input text to a max length of 1024 tokens due to the carrying capacity of the BART model. Encoding-decoding schemes of Transformers model revolutionaries the generation of summary perhaps the lay summary in research.

At first, most pre-trained language models focus on pre-training in general plain corpora from the Internet, like Wikipedia. The dataset needs to be cleaned with a very particular way may differ for each type of task and model. One another pre-trained model PEGASUS used by Zhang et al, 2019b [8] considered only abstracts to produce lay summary.

B. Approaches for Lay summarization in other domains

Another approach in generating the summary as prescribed in [1] is to generate an extractive summarization over the dataset which is then used to condition the transformer language

model on relevant information before being tasked with generating summary, authors did then summarization for more than one datasets ,e.g., PubMed, bigPatent. For Single longer-form documents, a hierarchical encoder that models the discourse structure of a document and an attentive discourse-aware decoder to generate the summary is used in [2]; authers extends RNN encoders to a hierarchical RNN, in decoder side authors used coverage vectors attention coverage to avoid repeatedly attending same steps. The presented Sequence2Sequence(S2S) model is able to effetely summarize long and structured documents while leaving room for still improvements.

IV. System Design:

A. BART ARCHITECTURE

Several tasks such as question answering, summarization require the understanding of language, so it is very important for the model to read the entire text as a whole and get a good understanding of each token, this goal is achieved through interpret the text through bi-directional reading approach And this idea is incorporated by BERT (Bidirectional Encoder Representations from Transformers). Which is the first part of our BART model architecture which finds the finest representation of the input sequence the specialty of BERT is that it has an encoder that produces vector embeddings for each token and as well as an extra vector for sentence-level understanding, because of which decoder can learn from both. After that an auto-regressive model comes to play that can look into past data to predict the future

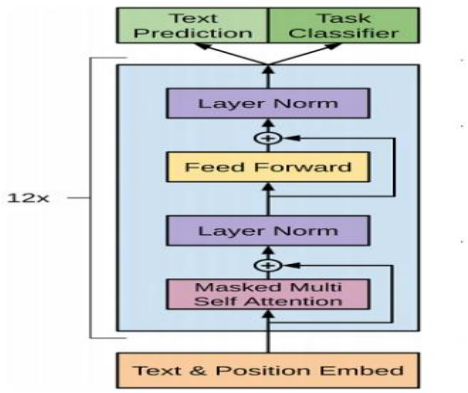

Figure 1: GPT architecture with 12 decoders stacked.

The GPT-1 model employed an architecture that resembled the decoder section of vanilla Transformers. To be precise, GPT has 12 such decoders that are arranged sequentially in a manner that learning from past tokens can influence the present token calculation. The model's structure is depicted above. The GPT decoder utilizes a masked multiheaded self-attention block and a feed-forward layer, which were also featured in the original Transformer decoder. BART also takes consideration of supervised approach for learning semi-supervised to be more specific. To compute the current token , the model pre-trained on 't' tokens looks back to 'k' tokens in the past.

$$L_1(T) = \sum_i log\, P(t_i|t_{i-k}, \ldots, t_{i-1}; \theta) \qquad (i)$$

Now already having feature vectors 'x', the model maximizes the probability of label 'y' on fine-tuning.

$$L_2(C) = \sum_{x,y} log\, P(y|x_1, \ldots, x_n) \qquad (ii)$$

Merging 1 and 2 gives as 3 , where lambda represents the parameters describing learned weights that controls the influence of language modeling.

$$L_3(C) = L_2(C) + \lambda L_1(C) \qquad (iii)$$

Thus in BART the bi-directional encoder is attached to the auto-regressive decoder that creates an architecture having denoising auto-encoder architecture, the encoders learns by looking at the entire input sequence using bi-directional approach, and decoder takes these learning's as an input to predict the next token.

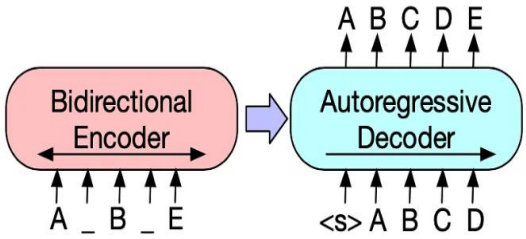

Figure 2: BART model

B. PEGASUS ARCHITECTURE

Similar to other tasks involving sequence transduction, PEGASUS employs the seq2seq architecture. However, what sets this architecture apart is its unique objective for self-supervised pre-training. It erases the need for labeled samples.

The process in PEGASUS involves taking sentences from a document and masking them, after which the model is trained to predict these sentences. This task is known as Gap Sentence Generation or GSG. The authors recognize that this task appears almost insurmountable, even for humans. Nonetheless, by engaging in such training, there is development in generating sentences that contain an instance of the original document, which underscores their assumption. Pegasus computes the ROUGE scores to choose the best sentence for masking; the author had found very good results by applying this method.

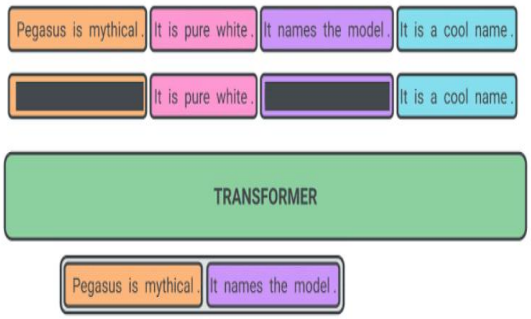

Figure 3: Gap Sentence Generation (GSG)

Other than GSG the encoder is pre-trained as a masked language model (MLM), for that words are randomly masked and the unmasked words are used to predict these masked words.

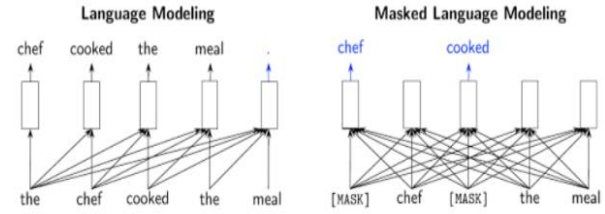

Figure 4: Masked language modeling (MLM)

After that both the methods are combined and the transformer is made to be trained in an incorporated manner.

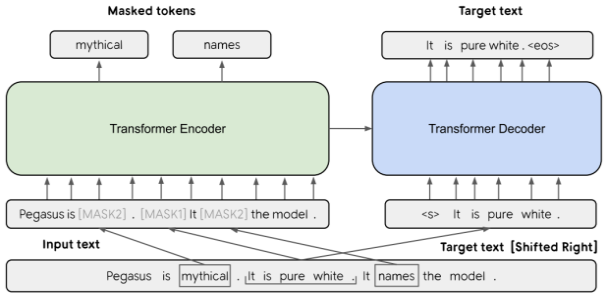

Figure 5: Pegasus Architecture

C. T5 ARCHITECTURE

Its transformer based architecture with some modifications, such as it uses the complete encoder-decoder architecture which is proven to be better than only using decoder. This requires an input and output pair to be used for fine-tuning and training the T5 model.

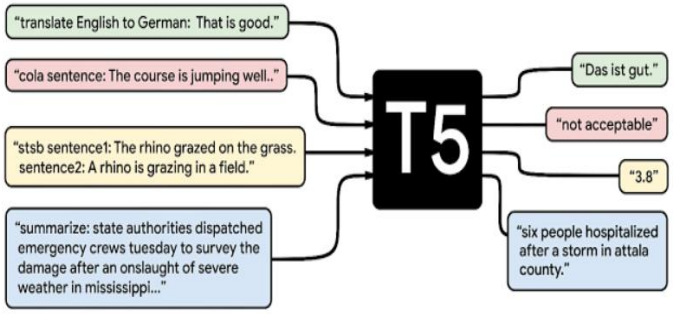

Figure 6: T5

Transformer architecture having left part as encoder and right part as decoder, T5 is having similar architecture like this one:

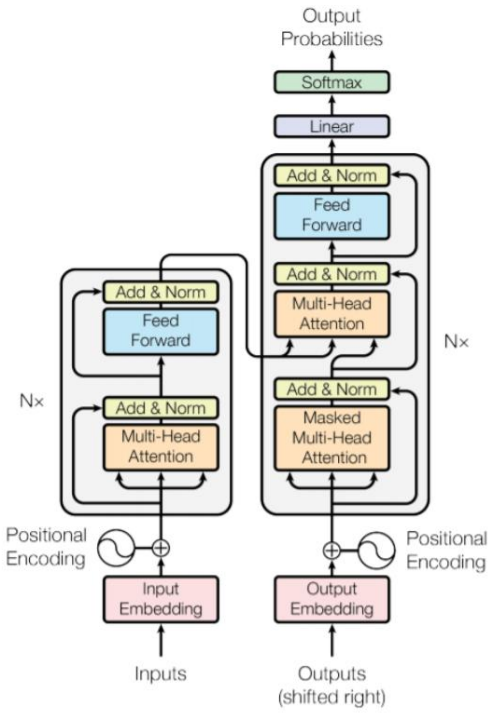

Figure 7: Transformer architecture

The author considered three variants of the transformer architecture for T5 model

1. Encoder-Decoder

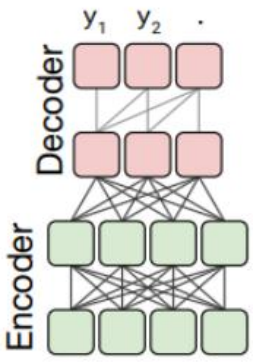

Figure 8: Encoder-decoder model

The usual approach for encoder-decoder architecture involves masking in the encoder and encoder-decoder attention, which is completely visible. On the other hand, the decoder uses causal masking or an attention mask. Masking is employed to ensure that the output at a certain position does not take into account future output while predicting.

2. Language model

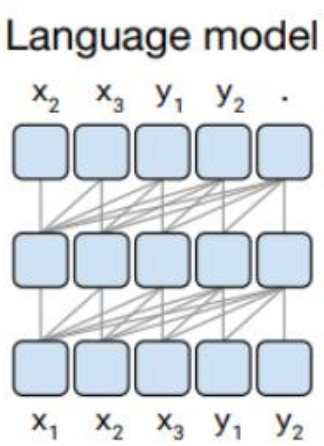

Figure 9: Language model

A language model includes a stack of Transformer layers and receives input and target inputs concatenated together, with a causal mask applied throughout. As is typical with language models, the output is only allowed to attend to the past input or output.

3. Prefix Language-model

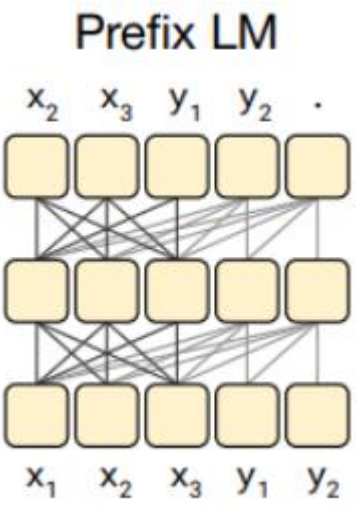

Figure 10: Prefix language model

When a prefix is added to a language model, it permits visible masking across a section of the input. This process is comparable to a regular

Language model, with the distinction being that the output focuses on a specific part of the input

that may contain task-specific details such as translating English to German.

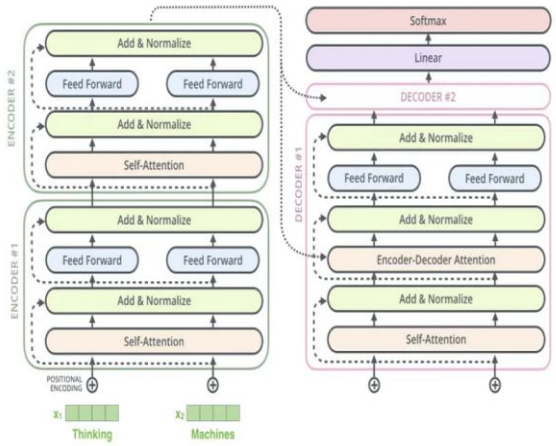

Figure 11: Final architecture of T5

V. PROJECT IMPLEMENTATION

A. DATA ANALYSIS:

We were provided with two datasets for our lay summarization task , the first one is PLOS dataset. This dataset contains 24773 medical articles including full-text, and abstract and their corresponding summaries and the other one is Elife dataset that also contains the similar contents as PLOS except it has only 4346 medical articles and their corresponding lay summaries. In the below tables we can see the average number of words and sentences in both the datasets.

Table 1: Avg. No. of words in and sentences in PLOS dataset

S.No.	Average	Articles	Summaries
1	Sentences	298.80	8.32
2	Words	6750.88	194.89

We can clearly notice from the table that there is much difference in terms of average number of words and sentences between PLOS dataset and ELIFE dataset , surely ELIFE here is dominating in terms of word and sentence count.

Table 2: Avg. No. of words in and sentences in ELIFE dataset

S.No.	Average	Articles	Summaries
1	Sentences	599.02	17.95
2	Words	10159.27	382.26

After that we analyzed the similarity between the medical articles and the lay summaries using different techniques such as JACCARD similarity and COMPRESSION similarity below we can see the similarity between articles and lay summaries in terms of scatter plots, only 500 documents similarities are shown below because plot of entire dataset was not clearly visible.

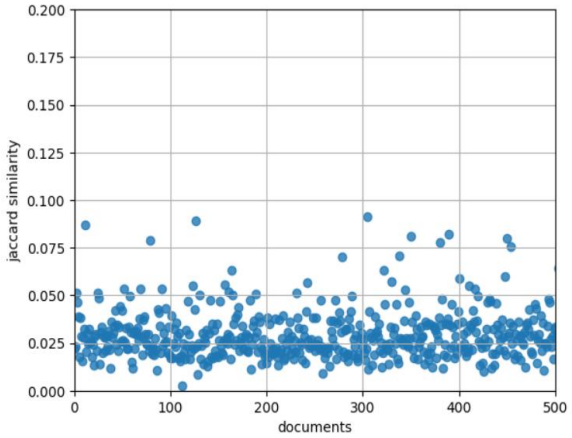

Figure 12: Jaccard similarity between articles and lay summaries in PLOS dataset

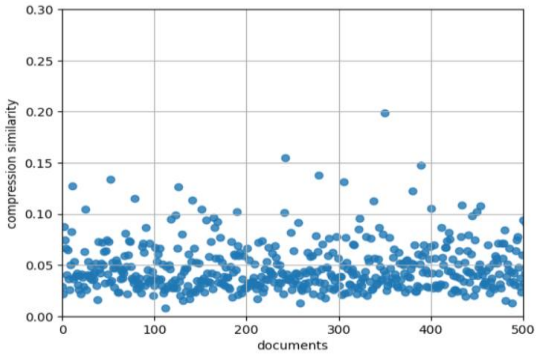

Figure 13: Compression similarity between articles and lay summaries in PLOS dataset

From the plot we can clearly conclude that there is not much similarity between the articles and lay summaries ,jaccard similarity did not even cross 10% mark, Below we can see the plots for ELIFE dataset

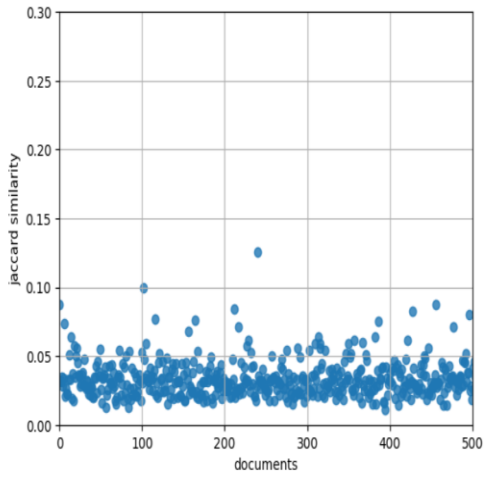

Figure 14: Jaccard similarity between articles and lay summaries in ELIFE dataset

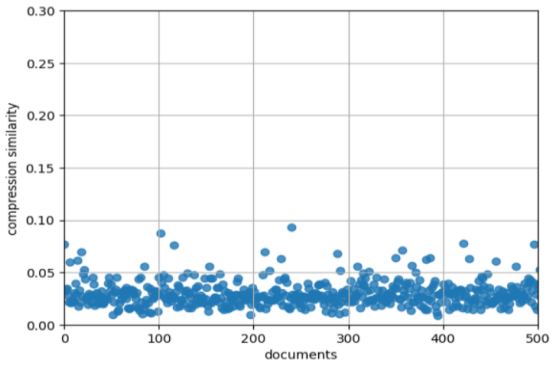

Figure 15: Compression similarity between articles and lay summaries in ELIFE dataset

In the below tables we can see the average similarity in the entire dataset.

Table 3: Avg. similarity between documents in both the datasets

S.No.	Similarity	PLOS	ELIFE
1	Jaccard	0.028	0.032
2	Compression	0.046	0.029

B. Data Preprocessing:

Once we realized the limitations of the unsupervised model, we decided to take a different approach. We manually created a list of commonly used acronyms in the biomedical domain, along with their expanded forms. We then used this list to replace any instances of these acronyms in the text with their expanded forms. Additionally, we also added the expanded form of the acronym in parentheses after the first instance of the acronym in the text to provide additional context to the reader.

After addressing the issue of acronyms, we moved on to the next step of data preprocessing which involved sentence splitting and tokenization. We used the NLTK library to split the text into individual sentences and then tokenize each sentence into its constituent words. This allowed us to feed the preprocessed text into our summarization models for training and testing. Overall, we found that our preprocessing steps greatly improved the performance of the summarization models on the lay summarization task.

Great to hear that you have preprocessed your data for better model performance! Using a Clinical Abbreviation Dictionary to expand acronyms seems like a sensible heuristic approach, given the specific domain of biomedical research articles. It's good to know that it performed well on your dataset, despite some limitations.

Moving forward with model design, have you considered using any specific architectures or algorithms for the task of lay summarization? It would be interesting to hear more about your approach and any considerations you had during the model design process.

C. Model Details:

We start by trying to emulate the results of the previous models developed by the researchers of our base paper. This will give us a good baseline to work on. For our heuristics-based approaches, we include the widely-used LEAD-3 baseline which simply uses the first three sentences of the main body of the text. As our lay summaries typically consist of more than three sentences, we also include LEAD- K, with k being equal to

the average lay summary length for each dataset.

For supervised models, we use the transformer-based BART base model, which we fine-tune on our own datasets. We used the base BART model which we trained for 5 epochs set to a learning rate of 2e-5. Looking at the dataset we realized that the average summary length is200 sowe feed this info to the model as well. We used cross entropy loss with label smoothing as it has been known to perform best for abstractive summarization tasks. Furthermore, we also fine-tuned PEGASUS and T5 models too. The design details were similar to the base BART model but each approach produces substantially different results.

The most remarkable results were produced when we fine-tuned BART not on the entire data but just 20% of the PLOS data. We were forced to take this approach due to initial hardware constraints which we later overcame. But surprisingly this model performed even better than the BART model trained on the entire data on the validation set. We feel like BART would lend itself perfectly for scarce data summarization tasks. We shall talk about the metrics used for evaluation of the models and the results obtained in the results section.

We also tried permuting with the amount of data per article that we passed. We passed the entire article (tokenized), passed just the abstract, abstract + introduction and abstract + introduction + conclusion. The best results were obtained by passing abstract + introduction. This observation is in keeping with our base paper and logically also make sense since the purpose of the abstract and introduction is to familiarize the reader with the topic.

D. HPC Implementation:

Local computing power wasn't enough to train such models; High Performance Computing [HPC] is required. SLURM is required to schedule jobs in HPC; the scheduling depends on configurations of HPC. Initially we trained the models using up to 4 nodes of CPU, with standard 256GB HPC RAM. Allocating 80G out of it with the following configurations:

Table 4: Configurations

Configurations	Values
Training Batch size	16
Train_Epochs	2
Learning Rate	2e-5
Seed	42
Validation batch size	8
Validation Epochs	1

But it failed to run for all 24K cleaned data, even 4 days weren't sufficient. We tried reducing the train batch size keeping everything constant but got no improvement. And decided to shift to GPU, we had 2 Nodes of 'gpu_v100' available. With little or no changes in training parameters and others as constant:

Table 5: Parameters

Model	Train Batch Size	Train Epochs	Validation Batch Size	Validation Epochs
Bart-base	32	2	8	1
Pegasus-pubmed	32	3	16	2
T5	8	2	4	1

And perhaps concluded 'Bart-base' turns out to be best results. So we proceeded with juggling the datasets, adding expanded abbreviations, etc. to give our final output.

VI. RESULTS

A. METRIC USED

We will now discuss the evaluation metrics that are used to measure how closely the model-generated summaries match the golden annotated summaries. It's worth noting that the ROUGE score has become the widely accepted standard evaluation measure for summarization tasks in the research community. ROUGE is abbreviation for Recall oriented understudy for

Gisting Evaluation, but other than recall it also considers Precision. ROUGE scores are of three categories ROUGE-1,ROUGE-2 and ROUGE-L

1. ROUGE-1

ROUGE-1 Precision and Recall evaluate how similar the unigram tokens are between the reference and candidate summaries. Unigrams refer to individual words used for comparison.

2. ROUGE-2

ROUGE-2 Precision and Recall measure the similarity of bigram tokens between the reference and candidate summaries. Bigrams consist of two consecutive words that are compared between the two summaries.

3. ROUGE-L

ROUGE-L Precision and Recall evaluate the longest common subsequence (LCS) of words between the reference and candidate summaries. LCS refers to word tokens that appear in a sequence, but not necessarily in consecutive order.

ROUGE-1, ROUGE-2, and ROUGE-L Precision/Recall metrics collectively provide an effective way to measure how well the model-generated summaries match the golden-annotated summaries. To simplify the scores further, F1 scores are often computed, which represent the harmonic mean between Precision and Recall for all ROUGE scores.

$$F1\ score = 2 * \frac{Precision\ * Recall}{Precision + Recall}$$

4. FKGL

Flesch-Kincaid Grade Level (FKGL) is a readability metric used in natural language processing (NLP) to estimate the grade level needed to understand a particular piece of text.

FKGL = 0.39 * (average words per sentence) + 11.8 * (average syllables per word) - 15.59

The resulting score represents the estimated grade level needed to understand the text, with higher scores indicating more difficult reading material. FKGL scores are often used in education, publishing, and other fields where it's important to ensure that written material is appropriate for a particular audience. NLP applications can use FKGL scores to help automate the process of determining the readability of text, which can be useful for content creators and educators.

5. BERT Score

The BERTScore is a metric that automatically evaluates the performance of text generation systems. Unlike other commonly used methods that assess syntactical similarity at the token level, the BERTScore prioritizes evaluating semantic similarity between the reference and hypothesis tokens. In tests conducted by the paper's authors on machine translations and image captioning, the BERTScore was found to be more closely aligned with human judgments.

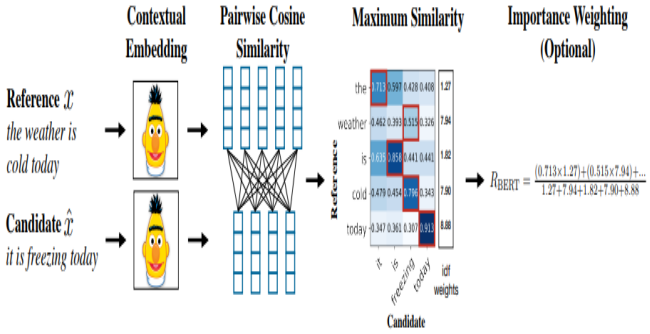

Figure 16: Computation of Recall metric

6. BART Score

BART score is a natural language processing (NLP) evaluation metric used to assess the quality of machine-generated text summaries. It is based on the BART (Bidirectional and Auto-Regressive Transformer) language model, which is a pre-trained neural network that can be fine-tuned for various NLP tasks.

To calculate the BART score, the BART model is used to generate a summary of a given input text, and then the generated summary is compared to a reference summary produced by a human. The BART score is then computed as the cosine similarity between the two summaries, taking into account the semantic meaning of the text.

BART score is designed to overcome some of the limitations of traditional evaluation metrics, such as ROUGE (Recall-Oriented Understudy

for Gisting Evaluation), which are based solely on surface-level features of the text, such as n-gram overlap. BART score takes into account the semantic similarity between the generated and reference summaries, which is more closely aligned with human judgments of quality.

B. RESULT TABLE:

As the best model we could come up with turns out to be 'Bart-base', following table shows the results with training different sets:

Table 6: Result Table

Training Dataset		Model used	Results
Description	Size	Description	Rogue-1 Score
PLOS	First 5000	Single Model trained only on PLOS	0.3540
PLOS + eLife (combined)	24,773 + 4,346 = 29,119	Single Model trained on combined datasets	0.3602
PLOS + eLife (seperate)	24,773 + 4,346	Two Bart-base different models, separately trained	0.3703

The above results are from the test dataset provided by the organizing committee. We have feed the preprocessed data as described above.

C. Conclusion:

To tackle this shared task problem statement, we applied various different models and a few novel data preprocessing techniques. All the models that we tried worked best after the data was cleaned and the abbreviation dictionary approach was applied. This gives credibility to such simplified heuristic cleaning approaches. Our best results were obtained by fine-tuning base BART on just 20% of the total PLOS data. Since the eLife dataset is very small in size, models trained specifically on this data performed poorly. But models trained by combining both eLife and PLOS performed best for the eLife verification set. But this was not true for the PLOS dataset. We think that this score can be improved by using different loss functions and training the model for different numbers of epochs.But to get really good results, the optimum approach would be to design and implement a custom architecture, specifically designated for this task. But for the lack of time and expertise we refrained from doing that. But the results we obtained, especially by training BART on limited data. Our results provide confidence for approaching summarization problems with low data. This may also be applicable to low data languages. We would love to work on such problems in the future.

REFERENCES

[1] Sandeep Subramanian, Raymond Li, Jonathan Pi- lault, and Christopher Pal. 2019. On extractive and abstractive neural document summarization with transformer language models. arXiv preprint arXiv:1909.03186.

[2] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Na-zli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. arXiv preprint arXiv:1804.05685.

[3] Benyou Wang, Qianqian Xie, Jiahuan Pei, Prayag Tiwari, Zhao Li, et al. 2021. Pre-trained language models in biomedical domain: A systematic survey. arXiv preprint arXiv:2110.05006.

[4] Yang Liu and Mirella Lapata. 2019. Text summariza tion with pretrained encoders. In Proceedings of the 2019 Conference on Empirical Methods in Nat ural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

[5] Haoyu Zhang, Jingjing Cai, Jianjun Xu, and Ji Wang. 2019a. Pretraining-based natural language gener ation for text summarization. In Proceedings of the 23rd Conference on Computational Natural Lan guage Learning (CoNLL), pages 789–797, Hong Kong, China. Association for Computational Lin guistics.

[6] Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. Discriminative adversarial search for abstractive summarization. arXiv preprint arXiv:2002.10375.

[7] Wonjin Yoon, Yoon Sun Yeo, Minbyul Jeong, Bong Jun Yi and and Jaewoo Kang. 2020. Learning by se mantic similarity makes abstractive summarization better. arXiv preprint arXiv:2002.07767.

[8] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Pe ter J Liu. 2019b. Pegasus: Pre-training with ex tracted gap-sentences for abstractive summarization. arXiv preprint arXiv:1912.08777

[9] Yu, Tiezheng, et al. "Dimsum@ laysumm 20: Bart-based approach for scientific document summarization." arXiv preprint arXiv:2010.09252 (2020).

[10] Hancock, John. (2004). Jaccard Distance (Jaccard Index, Jaccard Similarity Coefficient). 10.1002/9780471650126.dob0956.

[11] Vitányi, Paul. (2011). Compression-Based Similarity. Journal of Computer and System Sciences - JCSS. 111 - 118. 10.1109/CCP.2011.50.

[12] [BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension](https://aclanthology.org/2020.acl-main.703) (Lewis et al., ACL 2020)