PEGembed: Lay Summarization for Biomedical Scientific Journals

ZHIHENG WANG zhihengwang@umass
AHMED JAAFAR ajaafar@umass
JIARUI LIU jiaruil@umass
JORDAN PERRY-GREENE jperrygreene23@amherst

1 Problem Statement and Abstract

In a post-COVID world of breakneck research and even faster publication, a rare moment of public fascination is wilting as non-experts struggle to stay informed on current scientific developments. The fields of biology and medicine are particularly obtuse to non-experts in this context, with papers saturated with obscure terminology and requiring extensive background knowledge. These linguistic restrictions create a high overhead cost to learning, and to understand the detriment of such a price, we borrow an analogy from neuroscientist Sam Harris. Imagine the field of astronomy if each person had to build their telescope before they could even begin to see if astronomy was a legitimate enterprise; the sky wouldn't be any less worthy of investigation, but it would be tremendously more difficult to convince anyone to care about it.

Scientific papers hold detailed information about the newest advancements, but three main roadblocks stand between this source and general consumption: paywalls, time to read, and the aforementioned high overhead cost. While the first problem is beyond our scope, we attempt to address the latter two issues. We propose PEGembed, a tool to summarize biomedical papers in terms that are easier to understand. Through scientific lay summarization, we aim to start building a telescope anyone can use to stay engaged with scientific progress.

2 What you proposed vs. what you accomplished

- We planned to use both eLife and PLOS, but as it is way too expensive for us to compute with a reasonable budget, we discarded PLOS and decided to stick exclusively to eLife.
- We considered using some of the PreSumm models but ultimately decided against them for a couple of reasons. Any of the extractive models are immediately out of the question because our dataset did not have ground-truth extractive summaries. Additionally, their models were trained on news datasets, so their preproprocessing code was built specifically for that type of dataset. Since our dataset was medical and of a different structure, that meant some of that code was unusable. In place of PreSumm, we adopted a method proposed by Zhiheng Wang that employs the 'topK' cosine similarities for extractive summarisation. This approach aligns well with our dataset and provides effective results. Further details about this method and its implementation can be found in Section 6.1.
- We opted not to utilize the BERTSum for the extractive method, for reasons outlined in Section 9. Instead, we developed and implemented our own unique approach, and it is introduced in section 6.1.
- We proposed using COMET as one of our evaluation metrics, but instead chose to use METEOR. COMET was made to be used more for conversational tasks. METEOR on the other hand, is designed to capture semantic similarity and accuracy, which are crucial aspects in evaluating the quality of medical summaries. Its focus on content accuracy and semantic matching makes it more suitable for assessing the quality and alignment of medical article summaries.

3 Related work

The introduction of the attention mechanism and transformer model in the paper "Attention is all you need" introduced a new approach to text summarization. Since then, people are exploring to find a model that is able to summarise complicated datasets with easy-to-understand words. The TED model introduced in 2020 (Yang et al., 2020) is a method that is able to generate summaries accurately and concisely. TED has the potential to summarize biomedical papers, even though it is not trained specifically on those datasets.

Another reputable model, BERT, as introduced by Devlin et al. (Devlin et al., 2019) is a pre-trained bidirectional transformer designed for language understanding, allowing for easy fine-tuning for various applications. An excellent example of a BERT-based model is the BERTSum model (Liu, 2019), which has been particularly successful when fine-tuned on the CNN/Daily Mail dataset. These models have been increasingly utilized to summarise intricate papers within the biomedical field.

In 2021, Du (Du et al., 2020) proposed a strategy that leverages BERTSum. They fine-tune it on biomedical scientific papers to attain effective extractive summarization. The role of eLife, a leading dataset for biomedical papers, has been instrumental in this context. It provides a substantial corpus that researchers use for generating summaries and, going a step further, lay summaries.

A paper titled *Making Science Simple: Corpora for the Lay Summarisation of Scientific Literature* (Goldsack et al., 2022) highlighted the use of PubMed-BART, a pre-trained sequence-to-sequence model (Lewis et al., 2019). This model, when fine-tuned on scientific datasets, is capable of generating lay summaries for given papers, enhancing the accessibility of scientific literature. This paper made us aware that one requirement for summarization is that it should be easy to understand. Thus, it is worth it to look into some related work that focuses on "lay summarisation." In Seungwong Kim's paper *Using Pre-Trained Transformer for Better Lay Summarization"*(Kim, 2020), he uses the PEGASUS abstractive capabilities to generate lay summaries, and lengthens them by appending sentences extracted from the Presumm models (Liu and Lapata, 2019).

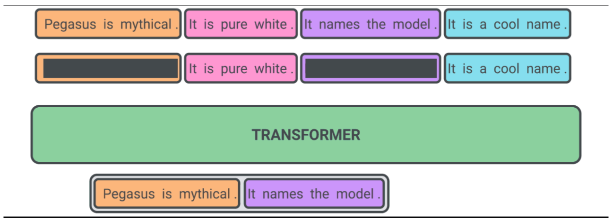


Figure 1: How PEGASUS works. Credits: Google

The PreSumm models utilize both abstractive and extractive summarization techniques. The abstractive model uses BERT as the encoder and a randomly-initialized Transformer for the decoder. The extractive model "is built on top of this encoder (BERT) by stacking several inter-sentence Transformer layers to capture document-level features for extracting sentences." The BERT used is not the original architecture, but rather BERT made specifically for summarization, called "BERTSUM."

BERTSUM outputs vectors for sentences rather than tokens. The extractive model is called BERTSUMEXT. The abstractive model is called BERTSUMABS. They experiment with the BERTSUMEXT and BERTSUMABS models separately and get results. Additionally, they combine these two models by first fine-tuning the encoder on the extractive task and then on the abstractive task. This two-stage approach uses extractive objectives to boost the abstractive summarization performance, utilizing information shared between the two tasks. This model is called BERTSUMEXTABS.

We seek to improve aspects of Kim's approach, in both the method of combination and more comprehensive evaluation metrics.

In Kim's approach above, we can see that researchers use abstractive summarisation to achieve the goal of "lay-summarization". PEGASUS is a state-of-the-art pre-trained transformer-based encoder-decoder abstractive summarization model with extracted gap sentences for abstractive summarization introduced by Zhang, Zhao, Saleh, and Liu (Zhang et al., 2020). PEGASUS masks entire sentences and asks the model to predict these "gap" sentences in the context of the remaining text, as seen in Figure 1. However, as discussed in (Phang et al., 2022), the asymmetry between the input length and summary lengths requires new considerations regarding the resource limitations of the models.

BigBirdPegasus (Zaheer et al., 2021) is a variant of PEGASUS. The main difference between BigBirdPegasus and Pegasus models is that BigBirdPegasus uses a sparse attention mechanism, while Pegasus does not. Sparse attention allows Big-

BirdPegasus to process longer sequences of text than Pegasus. Thus the model can process more for the same amount of memory. In addition, BigBirdPegasus also applies global attention as well as random attention to the input sequence, which further improves its performance on long-range summarization tasks.

One version of BigBirdPegasus is bigbird-pegasus-large-pubmed, which is a larger (more parameters) version of BigBirdPegasus that was fine-tuned on the PubMed dataset. At first, we fine-tuned this model on our dataset, eLife, but then realized that the PubMed dataset already includes eLife. This led us to abandon bigbird-pegasus-large-pubmed.

With the analysis above, we decide to develop our own tool mainly focused on biomedical paper, and use the combination of the extractive and abstractive methods, which will be shown below.

4 Your dataset

4.1 Dataset Introduction

eLife, a non-profit organization, aims to revolutionize scientific publishing by creating a platform for research communication that promotes ethical and responsible behaviors in science. Through its open-access journal, eLife publishes a wide spectrum of research, ranging from foundational theoretical work to applied and clinical studies in life sciences and biomedicine. Each article not only contains the original research but also includes an editor's summary, providing layman's term abstracts. These summaries serve as an excellent ground truth for abstractive summarization. Given our objective to develop a high-performing lay-summarization model for biomedical papers, eLife's comprehensive dataset, complete with research articles and their summaries, provides an ideal foundation for our project.

4.2 Dataset Structure

Our dataset comes from Goldsack et al.'s paper *Making Science Simple: Corpora for the Lay Summarisation of Scientific Literature* (Goldsack et al., 2022). Goldsack et al. construct a database containing the eLife dataset, with papers and its corresponding layman's term abstract. The dataset is stored in a JSON file.

```
{
  "id": "eLife-ids",
  "year": "publish date",
  "title": "title of the paper",
  "sections": "[[...]]",
  "headings": "[...]",
  "abstract": "[...]",
  "summary": "[...]",
  "keywords": "[...]"
}
```

Figure 2: JSON format

In the provided format, "sections" is a two-dimensional array where the outer array represents each paper and the inner array represents each section of the paper, excluding the abstract. Each inner array is a list of sentences in the respective section. Following that, "headings" is a list of the names of the sections in the paper, excluding the abstract. "abstract" is a list of sentences in the abstract section of the paper. "summary" is a list of sentences in the lay summary provided by the editor. Lastly, "keywords" is a list of tags for the paper. Regarding the data files, you mentioned the availability of "train.json," "val.json," and "test.json" in the eLife.zip file. However, for cost-related reasons, only "train.json" and "test.json" are utilized. Approximately 4331 papers from "train.json" were used for training and validation, while 241 papers from "test.json" were used for testing.

For extractive and abstractive summarization, we primarly use the "sections" field, while the "summary" field serves as the ground truth for evaluation. After we generate our own summary, we create a new "key: value" pair with key named as our summarization method and value as our summarization results.

5 Baselines

5.1 LeadK-Synonyms

LeadK as an extractive summarisation method is commonly used as a baseline in research focused on summarization. The basic premise of leadK is that the most crucial sentences typically appear at the beginning of the paper. To tailor leadK more effectively to our needs, we propose a modified version: leadK-Synonyms.

This method comprises two steps. The first step adheres to the traditional leadK approach. Our analysis of sentence distribution across paper sections suggested that selecting the top five sentences (if available) from each section provides an extensive coverage of the paper's content. In the second step, we utilize synonyms and hypernyms to replace less common words, referencing Princeton's WordNet (University, 2010). The motivation behind this step is to make our summary more accessible to non-specialist readers.

5.2 GPT-3.5

Our GPT-3.5 method is chosen as as an upper bound, and relies on OpenAI's model text-davinci-003. GPT-3.5 was picked as a massive and powerful state-of-the-art model with a very large token window of 4,097 tokens and cheap and fast API access. It fits our task better than the other three available GPT-3.5 models: text-davinci-002, code-davinci-002, and gpt-3.5-turbo for various reasons. text-davinci-002 was trained with supervised fine-tuning instead of reinforcement learning, limiting novelty and adaptability. code-davinci-002 and gpt-3.5-turbo are inferior for lay summarization as they are optimized for code and chat capabilities respectively. GPT-3.5's large token window is ideal, and even with such a large window nearly all papers exceeded this by a very large margin. To generate our summaries, we split the paper into chunks of "words" – the sets of characters that arise when we split on whitespace. We take 2500 words, rounded to the nearest sentence, with 2500 being experimentally determined to be the upper limit for the input size. We iteratively tested various prompts until identifying the following prompt as yielding suitable outcomes, letting chunk_text be the text from our 2500-word chunk and reply_summary be the concatenated summaries of previous chunks.

```
Briefly summarize the
following in simple terms:
\n{chunk_text}\n
Summary:{reply_summary}
```

This prompt is passed into text-davinci-003 with a temperature of 0.7. The discrepancy between text-davinci-003's maximum token allowance of 4097 and the upper limit of 2500 words is due to multiple factors. First, OpenAI's tokenization strategy is likely different than the simple whitespace split we employ. Second, additional tokens are required for the text included in every prompt. Finally, we require sufficient token space for all summaries of previous chunks to be appended to the prompt.

Our method of prompting GPT-3.5

5.3 Non-Fine-Tuned PEGASUS-X Baseline

To contextualize the performance of our fine-tuned PEGASUS-X models, we included the non-fine-tuned PEGASUS-X as a baseline in our comparisons. Given the specific demands of our summarization task—summarizing biological papers in the eLife dataset—we did not anticipate that the non-fine-tuned model would yield exceptional results.

Biological papers often contain highly specialized and technical language that requires a deep understanding of complex, interrelated biological processes and concepts. Furthermore, these papers typically adhere to a specific structure that may be challenging for a model to navigate without domain-specific fine-tuning. Hence, we used this non-fine-tuned model as a baseline to underscore the improvements offered by fine-tuning.

We utilized the non-fine-tuned models 'google/pegasus-x-base' and 'google/pegasus-x-large' directly from Huggingface's Transformers library. To generate output summaries, we passed our dataset through these models without any preliminary fine-tuning. We then evaluated the performance of these models using our own machine evaluation code, result can be found in section 7. This straightforward implementation allowed us to obtain baseline performance metrics, against which we could measure the effectiveness of our fine-tuning efforts.

6 Our approach

Motivated by the paper written by Kim Seungwon (Kim, 2020) which was discussed in the related work section, we decided to develop our own method by trying extractive summarization first then abstractive summarization.

The underlying premise of this approach is that a long extractive summary, even of this limited length, should encapsulate the majority of the critical information within the original document. Consequently, we hypothesize that these summaries should provide a sufficient foundation from which to generate reasonable and contextually accurate abstractive summarizations. This methodology allows us to leverage the strengths of our model while remaining cognizant of our re-

	xbase1	xbaseFull	xLarge1	xLargeFull
#Params	272M	272M	568M	568M
Batch Size	5	5	5	5
Block Size	1024	1024	1024	1024
learning rate	4e-5	4e-5	5e-5	5e-5

Table 1: Hyperparameters of our fine-tuned model

source limitations.

Operating within the constraints of a limited budget and computing resources, we have strategically tailored our approach to extractive summarization. We have set a cap of approximately 800 words for the extractive summaries to balance information retention with computational efficiency in fine-tuning step.

Subsequently, these long extractive summaries serve as input for fine-tuning our abstractive summarization model. The ground truth for this stage is provided by human-written summaries from the dataset. (The result part is offered in section 7.1)

Unfortunately, we won't be able to compare our results with theirs because their dataset is different from ours. There won't be consistency for us to be able to do comparisons.

6.1 Extractive Section:

The extractive phase of our methodology is critical, yet challenging due to the absence of ground truth for extractive summarization in our dataset. Given the scale of our dataset, generating this human-written ground truth is impractical.

Nevertheless, we have designed robust extractive methods that provide beneficial inputs for the abstractive phase. Preliminary evaluations indicate promising outcomes for the abstractive summaries generated from our extractive summaries.

6.1.1 Top K Sentence Cosine: the method that we used

Our first proposal for extractive summarisation draws inspiration from Reimers and Gurevych's work (Reimers and Gurevych, 2019). The author proposed a method to obtain sentence embeddings by modifying the pre-trained BERT model. These sentence embeddings can be compared using cosine similarity to find sentences with similar meanings. We used this to obtain the most important sentences in the paper, which leading to the development of the "Top K Sentence Cosine" method. This approach hinges on the computation of sentence embeddings for every sentence in a given scientific paper, employing the "all-MiniLM-L6-v2" model for this purpose. This model maps sentences & paragraphs to a 384 dimensional dense vector space and can be used for tasks like clustering or semantic search. However, as stated before, we failed to find any the ground truth for the extractive summarisation on any biomedical dataset, thus, we are unable to fine-tune the sentence-bert model. Therefore, our sentence-bert used default values. Luckily, our generation proved to be effective in later processes.

We then compute cosine similarities between these embeddings, aiming to identify sentences with semantically similar meanings. Our underlying hypothesis is that the importance of a sentence is directly proportional to its frequency of occurrence (albeit in varied forms) within the paper. In essence, sentences encapsulating the most significant insights are likely to be paraphrased or reiterated in some form, and their corresponding words should hold substantial weight within the overall paper.

To generate an extractive summary for each paper, we select the top 'k' sentences with the highest cosine similarities, while retaining their original sequence from the text. This ensures that our summarisation captures the most critical content and maintains its contextual continuity.

Balancing between the computational cost and summarisation efficacy (we are using A100 GPUs from colab), we set 'k' to 20 for our experiments. Given that the average number of sentences in a paper is roughly 300 (we ran our code from train.json and test.json), our choice of 'k' encapsulates around 10% of the total input length. This proportion provides a satisfactory and cost-effective coverage for the extractive summarisation stage, efficiently distilling the most relevant and insightful content from each paper.

6.2 Abstractive Summarization Method: Pegasus-X

In our work, we opted to utilize PEGASUS-X (Phang et al., 2022) (an extension of PEGASUS) as our abstractive summarization model for several compelling reasons. Firstly, PEGASUS has demonstrated superior performance across a multitude of datasets and summarization tasks, showcasing its proficiency in generating high-quality, human-like summaries. Secondly, PEGASUS-X offers a significant improvement over its predecessor by having the capacity to handle inputs of up to 16K tokens, compared to the original PEGASUS's limit of 512 tokens. Lastly, one of the key strengths of PEGASUS is its ability to achieve acceptable performance without requiring an extensive number of examples for fine-tuning, which is wonderful with our limited resources.

Id	Model	# examples
xbase1	PEGASUS-X_{Base}	2000
xbaseFull	PEGASUS-X_{Base}	3515
xLarge1	PEGASUS-X_{Large}	2000
xLargeFull	PEGASUS-X_{Large}	3515

Table 2: the 4 model we fine-tuned

Guided by these factors, we undertook the process of fine-tuning 2 distinct versions of the PEGASUS-X model each utilizing 2 different number of training examples. The fine-tuning process remained consistent across all four models. The model-specific hyperparameters for fine-tuning can be found in below table 1.

We initially fine-tuned the PEGASUS-X_{Base} model, labeled as 'xbase1/Pegasus_X_base_2k', using 2000 examples of the long extractive summary we got from the **TOP K Sentence Cosine** method. Following this, we further expanded the training examples to 3515 for a second PEGASUS-X_{Base} model, referred to as 'xbaseFull/Pegasus_X_base_4k'.

Turning our attention to the larger PEGASUS-X_{Large} architecture, we once again started with a set of 2000 examples to fine-tune a model we labeled 'xLarge1/Pegasus_X_large_2k'. Subsequently, we fine-tuned a second PEGASUS-X_{Large} model, 'xLargeFull/Pegasus_X_large_4k', using the larger dataset of 3515 examples.

This methodical approach was adopted to investigate the influence of varying parameters and training data quantities on the performance of the models. Both sets of training data consisted of long extractive summaries paired with human-written ground truths, drawn from half of the elife dataset and 81 parent of the elife datasets respectively.

Our choice of hyperparameters was primarily guided by our computational resources and the characteristics of our dataset. We had access to an A100 GPU with 40G of memory for the PEGASUS-X_{Base} model and an A100 GPU with 80G of memory for the PEGASUS-X_{Large} model at AutoDL platform. Given these resources, we set the batch size to 5 for both models to ensure that the computations fit within the available memory.

The block size was set to 1024 tokens. This decision was made based on our hypothesis that an extractive summary of approximately 800 words would be sufficient to capture the most important information from the original paper. A block size of 1024 also helped us maintain a balance between summary detail and computational efficiency.

Considering the size of the eLife dataset we were working with, we decided to use a lower learning rate than that used in the original PEGASUS-X paper. The original paper utilized a learning rate of 8e-4, but given the relatively small size of our dataset, we opted for rates of 5e-5 and 4e-5 instead. This decision was made to prevent the potential for overfitting that could arise from using a higher learning rate with a smaller dataset.

6.3 Resource used

Our fine-tuning process leverages the models available in Huggingface's Transformers library, a well-regarded resource in the NLP community for its efficient and effective transformer-based models. This library provided us with a solid foundation for our work with the PEGASUS-X models.

To craft our fine-tuning script, we took inspiration from the work of @jiahao87. His script, 'pegasus_fine_tune.py', served as a valuable starting point for our own fine-tuning process. The script, publicly available at `https://gist.github.com/jiahao87/50cec29725824da7ff6dd9314b53c4b3`, provided a roadmap for implementing our own fine-tuning procedures with the PEGASUS-X models.

Our work used a diverse range of computing resources for different stages of the project. For the fine-tuning and evaluation of the abstractive summarization models, we relied on the AutoDL platform. We utilized an A100 GPU with 40G of memory for the PEGASUS-X_{Base} model and an A100 GPU with 80G of memory for the PEGASUS-X_{Large} model.

For the Top K Sentence Cosine extractive summarization method, we employed an A100 GPU with 40G of memory GPU available on Google Colab. A locally-hosted RTX 3080 GPU and Google Colab's T4 GPU were also instrumental in data formatting tasks.

Finally, for the evaluation of all base models, we utilized an A100 GPU with 40G of memory at Google Colab. This diverse and flexible combination of resources was vital in allowing us to efficiently and effectively manage the various computational demands of our project.

7 Error Analysis

Given the context of text generation, it is prudent to thoroughly examine both machine evaluation and human evaluation. The inclusion of human evaluation is particularly valuable due to the current lack of a robust mechanism for assessing machine-generated paragraphs or passages at a comprehensive level (Karpinska and Iyyer, 2023). By incorporating both perspectives, a more comprehensive and insightful assessment of the generated text can be attained.

7.1 Machine Evaluation

7.1.1 Metrics

Our deliberations have led us to the adoption of a tripartite approach, employing three distinct metrics, namely ROUGE (Lin, 2004), BERTScore (Zhang et al., 2020), and METEOR (Banerjee and Lavie, 2005). This selection represents a judicious fusion of both non-learning-based and learning-based metrics, rendering it particularly well-suited for the task of summarization.

ROUGE, an established evaluation measure, has garnered significant recognition within the natural language processing community. It leverages overlap between the generated summary and reference summaries, employing a variety of n-gram co-occurrence statistics to assess the quality and effectiveness of the summary. Specifically, we utilized ROUGE-1, ROUGE-2, and ROUGE-L. The reason we used the first two is due to them being a standard, often used anytime ROUGE is used. On the other hand, ROUGE-L, which stands for Longest Common Subsequence (LCS), measures the longest subsequence of words shared by the generated summary and the reference summaries. It accounts for word order and sentence structure, making it particularly well-suited for capturing the summary's overall coherence and capturing the main ideas expressed in the source text. ROUGE-L is especially beneficial in summarization tasks where maintaining the original meaning and organization of ideas is crucial.

BERTScore, on the other hand, is a state-of-the-art metric that capitalizes on contextualized word embeddings. By utilizing contextual information, BERTScore can capture semantic similarities between the generated summary and the reference summaries with greater accuracy, resulting in a more nuanced evaluation.

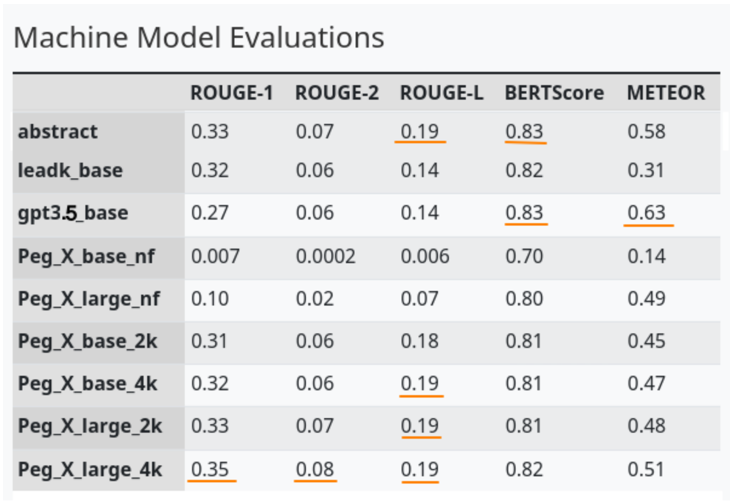
Lastly, METEOR offers a unique approach to summarization evaluation by incorporating a combination of precision, recall, and alignment-based metrics. It incorporates a rich set of linguistic and semantic resources, enabling a comprehensive assessment of the generated summary's overall quality.

The choice to employ this trichotomy of metrics stems from our desire to obtain a holistic and multidimensional evaluation of the summarization task. By leveraging both traditional statistical approaches and cutting-edge learning-based techniques, we aim to capture diverse facets of summary quality, thereby facilitating a more comprehensive analysis and interpretation of the generated summaries.

7.1.2 Analysis

As depicted in Figure 3, the comprehensive evaluation table showcases the inclusion of metrics at the top, while the models, alongside the abstracts, are presented along the left axis. It may raise curiosity as to why the abstracts are included in the evaluation. The rationale behind this decision stems from the fact that an abstract inherently serves as a condensed summary of the corresponding paper. Consequently, considering the abstracts for comparison purposes contributes to a more comprehensive and exhaustive assessment.

Among the metrics employed, namely BERTScore and METEOR, it is noteworthy that the baselines exhibited superior performance compared to our fine-tuned models. Notably, the margin of victory for BERTScore was minimal, with PEGASUS-X-Large-4k narrowly trailing


Machine Model Evaluations

	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore	METEOR
abstract	0.33	0.07	0.19	0.83	0.58
leadk_base	0.32	0.06	0.14	0.82	0.31
gpt3.5_base	0.27	0.06	0.14	0.83	0.63
Peg_X_base_nf	0.007	0.0002	0.006	0.70	0.14
Peg_X_large_nf	0.10	0.02	0.07	0.80	0.49
Peg_X_base_2k	0.31	0.06	0.18	0.81	0.45
Peg_X_base_4k	0.32	0.06	0.19	0.81	0.47
Peg_X_large_2k	0.33	0.07	0.19	0.81	0.48
Peg_X_large_4k	0.35	0.08	0.19	0.82	0.51

Figure 3: Machine Evaluation Metrics. Underlined are the best models for each metric."nf" ="non finetuned"

by a mere 0.01. The suboptimal performance of the baselines could potentially be attributed to inadequate hyperparameter tuning of the BERTScore model. Furthermore, in the case of BERTScore, we might have used an inferior BERT model for our task, RoBERTa, when instead we could have maybe used a different one such as bert-base-uncased for example.

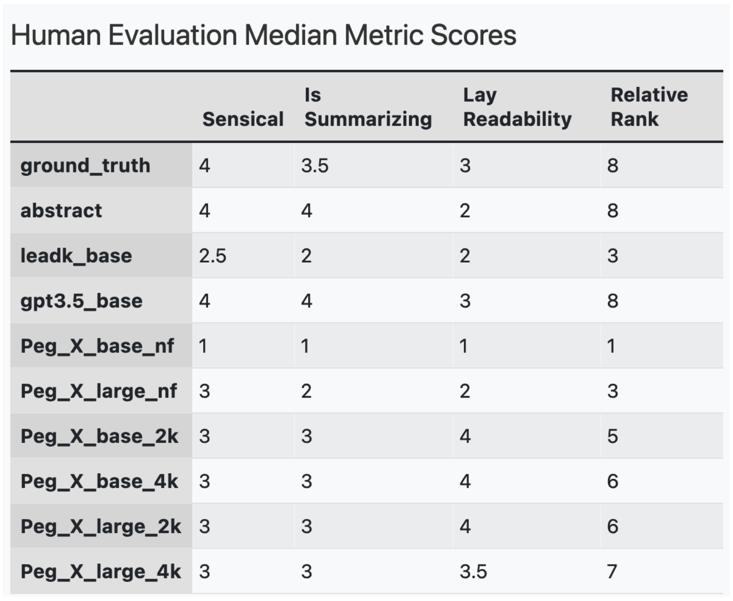
On the other hand, our fine-tuned models either matched or surpassed the baselines in three out of the five metrics, namely ROUGE-1, ROUGE-2, and ROUGE-L, with ROUGE-1 exhibiting the most significant discrepancy. One plausible explanation for the superiority of our fine-tuned models in terms of ROUGE lies in the abstractive nature of PEGASUS. Theoretically, PEGASUS has the propensity to generate a greater number of commonly used terms within its summaries. It may be argued that GPT-3.5, too, possesses the capacity to generate such terms. However, a distinguishing factor contributing to our models' triumph, particularly over a baseline like GPT-3.5, is the fine-tuning process. By customizing our models on medical data, they acquired the ability to generate medical terms that align more closely with the reference summaries. Consequently, more of the terms in the reference summaries would be in the generated summaries, thus resulting in higher ROUGE scores. Additionally, it is logical that the abstract achieved a high ROUGE score, given that, by definition, it incorporates terms and words extracted directly from the corresponding paper.

The poorest-performing model by a substantial margin was PEGASUS-X-Base-Non-Finetune. This can be attributed to its inherent nature as a base model, its relative scare number of parameter, coupled with the absence of fine-tuning on our medical dataset.

Conversely, the most exemplary model in terms of performance emerged as PEGASUS-X-Large-4k, which can be attributed to several factors. Firstly, it boasts the highest number of parameters, thereby endowing it with enhanced expressive capabilities. Secondly, its training data encompassed a more extensive corpus, consisting of approximately 4,000 papers. Lastly, and perhaps most crucially, it underwent a meticulous fine-tuning process tailored specifically to our medical dataset. Collectively, these factors contributed to its remarkable performance and overall superiority among the evaluated models.

7.2 Human Evaluation

Ten models were scored across 40 papers and evaluated on each of the four metrics: three scalar, and one ranking. Figures 4 and 5 display the average scores for each model on our scalar and ranking metrics in terms of the mean and median. The first major note is that the ground-truth summary, the abstract, and the GPT-3.5 baseline are virtually identical, with the only meaningful difference being a drop in lay readability in the abstract. This drop is expected as the abstract generally summarizes for an expert audience, and contains technical terms that explored in the paper without using space to explain them. The similarity between these three is striking, and the fact that human evaluators preferred these options nearly equivalently is a testament to both the inherent summarative quality of the abstract and the power of the


Human Evaluation Median Metric Scores

	Sensical	Is Summarizing	Lay Readability	Relative Rank
ground_truth	4	3.5	3	8
abstract	4	4	2	8
leadk_base	2.5	2	2	3
gpt3.5_base	4	4	3	8
Peg_X_base_nf	1	1	1	1
Peg_X_large_nf	3	2	2	3
Peg_X_base_2k	3	3	4	5
Peg_X_base_4k	3	3	4	6
Peg_X_large_2k	3	3	4	6
Peg_X_large_4k	3	3	3.5	7

Figure 4: Human Evaluation Scores (median). “nf” =“non finetuned”

GPT-3.5 method we employed.

A notable outlier of the poorer baselines is the non-fine-tuned xBase model, which scored the lowest score in *every metric, in every human evaluation*. This was a result of its ”summaries” being unreadable messes of punctuation and/or a simple repetition of the same word. Lead-K and non-fine-tuned xLarge performed better, but found poor metrics across the board, suffering especially in Lay Readability and Is Summarizing respectively. Lead-K’s poor performance in Lay Readability is likely due to the complex terms throughout the original papers. The non-fine-tuned xLarge model struggles with a deficit in grammar, and combined with its poor understanding of the task results in poor summarative capabilities. This model also tended to start with decent summary content, but then devolve into repeating variations of a similar sentence or sentence fragment until it returned.

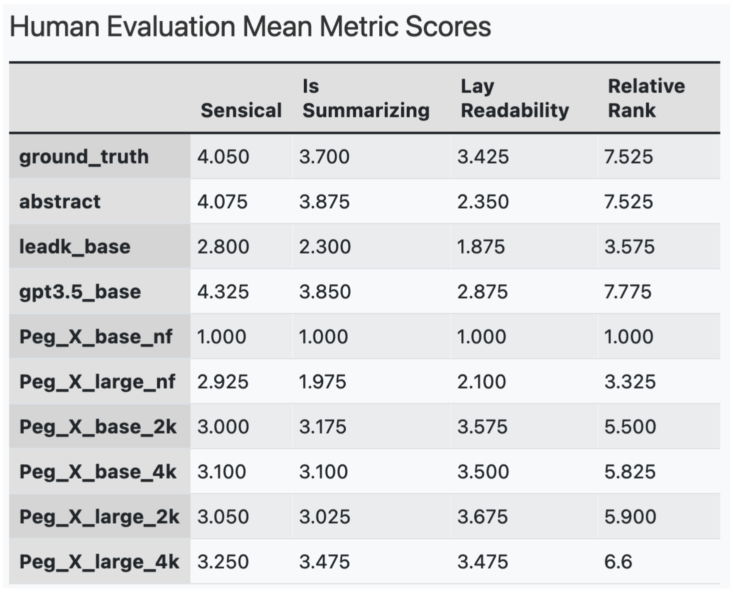
More interesting is the performance of our fine-tuned models. Increasing the fine-tuning set and boosting the model size from Base to Large improved performance both for Sensical and Is Summarizing scores, but made little change to Lay Readability. Human evaluators vastly preferred the xLarge_4k model to the other fine-tuned models. While not reaching the same sensical capabilities as the peak baselines, the fine-tuned PEGembed models stayed competitive in terms of summarative ability and drastically outperformed other models in terms of lay readability. Additionally, human evaluators consistently favored all fine-tuned PEGembed models over the baselines of Lead-K and their non-fine-tuned counterparts.

7.2.1 Evaluator Guidelines

Summaries will be evaluated based on three distinct scales: **Sensical**, **Is Summarizing**, and. **Lay Readability**.

- **Sensical Scale:**
  1: Unreadable/incomprehensible.
  2: Contains fragments of correct grammar or sentences, but very hard to parse meaning.
  3: Sections of readable, directed text, but contains major, obvious errors.
  4: Multiple sections of readable, directed text, but contains minor errors in grammar or meaning.
  5: Excellent, error-free grammar.

- **Is Summarizing Scale:**
  1: Replacement for N/A, i.e., grammar makes it basically impossible to understand what’s going on.
  2: Not trying to summarize/looks like a straight extract from the original paper.
  3: Condenses a concept, but either goes into detail or makes no effort to summarize elsewhere in the summary.
  4: Makes a meaningful effort to summarize throughout.



Human Evaluation Mean Metric Scores

	Sensical	Is Summarizing	Lay Readability	Relative Rank
ground_truth	4.050	3.700	3.425	7.525
abstract	4.075	3.875	2.350	7.525
leadk_base	2.800	2.300	1.875	3.575
gpt3.5_base	4.325	3.850	2.875	7.775
Peg_X_base_nf	1.000	1.000	1.000	1.000
Peg_X_large_nf	2.925	1.975	2.100	3.325
Peg_X_base_2k	3.000	3.175	3.575	5.500
Peg_X_base_4k	3.100	3.100	3.500	5.825
Peg_X_large_2k	3.050	3.025	3.675	5.900
Peg_X_large_4k	3.250	3.475	3.475	6.6

Figure 5: Human Evaluation Scores (mean). “nf” =“non finetuned”

5: Presents condensed and concise information.

- **Lay Readability Scale:**

  1: Not many readable words, Replacement for N/A, i.e., not many real words.

  2: Uses lots of complex language. Essentially the original paper using full complex terms.

  3: Contains several complex terms, but not an excessive amount.

  4: Few complex terms, or meaningful attempt at context and explanation for the majority of the complex terms used.

  5: Complex terms minimized, with context given for the ones used.

In each scale, a higher number equates to a better summary. Summaries should aim for a ’5’ in each category for the highest quality.

All summaries for a given paper will be compared to each other in a **ranking categorization** as well. Rank the following summaries based on how well an understanding of the original paper they conveyed to you, a non-expert. Traits such as the ones in the scalar categories above should be considered. Additional considerations should include how much you feel the summary respected your time and explained clearly and concisely, as well as whether it went into uninteresting or unnecessary detail. This ranking is intended as an overall assessment of quality, and your answer need not rely on the numbers you provided above. Rank favorable summaries as a higher number than unfavorable summaries.

7.3 Guideline Reasoning

The guidelines are selected for the overall goal of judging quality of a lay summarization. ”Sensical” is created as an assessment of grammar and cohesion, to compare the model to human-written text that isn’t necessarily summarative. ”Is Summarizing” is intended to gauge how much of a summary it is, while ”Lay Readability” is meant to judge how approachable that summary is for a layperson. The final ranking was performed across the ten summaries of a given paper, and was meant to be a subjective preference towards a given summary. While the first three were intended towards drawing objective conclusions about the quality of the summaries in a vaccuum, the final ranking was meant to relate the papers to each other.

8 Contributions of group members

List what each member of the group contributed to this project here. For example:

- Zhiheng Wang:
  1. Research on extractive summarization (bertSum, sentence-bert, leadKSynonym)
  2. write
     (a) Section 2
     (b) Section 3 before preSumm

    (c) Section 4, find the dataset and modify the dataset to our use
    (d) Section 5.1
    (e) Section 6 before 6.2
    (f) README.md on github
    (g) Section 9.1
  3. all the extractive code

- Ahmed Jaafar:
  - Writing
    * Some parts of the Related Works section.
    * Some parts of the AI Disclosure section
    * A large portion of the Machine Evaluation section
    * Machine Model Evaluations table.
    * Some parts of the Proposal vs Accomplished section
    * Some parts of the Conclusion section
    * A bit of the Your Approach section
    * PEGASUS image
  - Summary generation (decoding) code on the held-out test set
  - Deciding which of the evaluation metrics to use for our final product
  - Machine evaluation code
  - Generating some of the machine evaluation code
  - Looking into and attempting to get PreSumm code working
  - Code to make CSV of all our models and their respective metric scores
  - Human evaluation on 10 examples

- Jiarui Liu:
  1. Research on abstractive summarization (specifically, PEGASUS)
  2. write
     (a) section 5.3 non-fine-tuned Pegasus-X Baseline
     (b) part of introducing at section 6 ”Your approach”
     (c) section 6.2 ”Abstractive summarization Method: Pegaus”
     (d) section 6.3 ”Resource used”
  3. write our fine-tuned script for Pegasus-X and some data formatting code
  4. fine-tune 4 Pegasus-X model on server and do the evaluation with code from Ahmed

- Jordan Perry-Greene:
  1. Research:
     (a) prior methods of combining summarization methods with each other
     (b) evaluation methods and metrics
  2. write
     (a) Problem Statement and Abstract
     (b) Some parts of Related Work
     (c) GPT-3.5 baseline
     (d) Human Evaluation Error Analysis
     (e) Evaluator Guidelines
     (f) Guideline Reasoning
  3. All GPT-3.5 baseline code
  4. All human evaluation code

‘

9 Conclusion

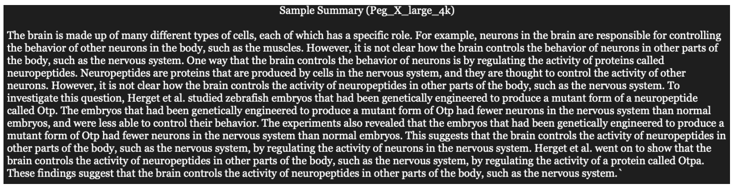
9.1 Discussion

Given the expansive dataset associated with our project, processing it presents significant computational and financial challenges. Pre-processing is particularly challenging because we lack specific knowledge for tokenization, which is critical for biomedical papers. We’ve attempted to find guidance online, but most resources have not proven helpful due to our lack of a biological background.

We opted for the LeadK method as our baseline approach. This traditional method serves as a reliable baseline for extractive summarization. We augmented the summaries with synonym replacements for uncommon words to make them more accessible, though as expected, the performance was modest. As an additional layer, we utilized GPT for abstractive summarization, which yielded excellent results.

For extractive summarization, we devised our own cost-effective method: topK Sentence Embedding Cosine Similarity. This method offers a feasible alternative to the BertSum model.

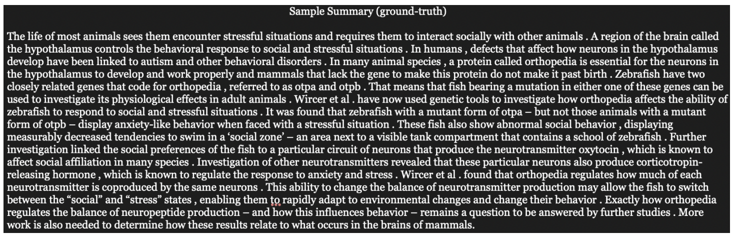
Although we attempted to use the BertSum model for extractive summarization on 750 results from train.json, the cost per paper was considerably higher than our in-house method. Furthermore, initial test results showed minor differences



Sample Summary (Peg_X_large_4k)

The brain is made up of many different types of cells, each of which has a specific role. For example, neurons in the brain are responsible for controlling the behavior of other neurons in the body, such as the muscles. However, it is not clear how the brain controls the behavior of neurons in other parts of the body, such as the nervous system. One way that the brain controls the behavior of neurons is by regulating the activity of proteins called neuropeptides. Neuropeptides are proteins that are produced by cells in the nervous system, and they are thought to control the activity of other neurons. However, it is not clear how the brain controls the activity of neuropeptides in other parts of the body, such as the nervous system. To investigate this question, Herget et al. studied zebrafish embryos that had been genetically engineered to produce a mutant form of a neuropeptide called Otp. The embryos that had been genetically engineered to produce a mutant form of Otp had fewer neurons in the nervous system than normal embryos, and were less able to control their behavior. The experiments also revealed that the embryos that had been genetically engineered to produce a mutant form of Otp had fewer neurons in the nervous system than normal embryos. This suggests that the brain controls the activity of neuropeptides in other parts of the body, such as the nervous system, by regulating the activity of neurons in the nervous system. Herget et al. went on to show that the brain controls the activity of neuropeptides in other parts of the body, such as the nervous system, by regulating the activity of a protein called Otpa. These findings suggest that the brain controls the activity of neuropeptides in other parts of the body, such as the nervous system.`

Figure 6: PEGembed sample summary.



Sample Summary (ground-truth)

The life of most animals sees them encounter stressful situations and requires them to interact socially with other animals . A region of the brain called the hypothalamus controls the behavioral response to social and stressful situations . In humans , defects that affect how neurons in the hypothalamus develop have been linked to autism and other behavioral disorders . In many animal species , a protein called orthopedia is essential for the neurons in the hypothalamus to develop and work properly and mammals that lack the gene to make this protein do not make it past birth . Zebrafish have two closely related genes that code for orthopedia , referred to as otpa and otpb . That means that fish bearing a mutation in either one of these genes can be used to investigate its physiological effects in adult animals . Wircer et al . have now used genetic tools to investigate how orthopedia affects the ability of zebrafish to respond to social and stressful situations . It was found that zebrafish with a mutant form of otpa – but not those animals with a mutant form of otpb – display anxiety-like behavior when faced with a stressful situation . These fish also show abnormal social behavior , displaying measurably decreased tendencies to swim in a ‘social zone’ – an area next to a visible tank compartment that contains a school of zebrafish . Further investigation linked the social preferences of the fish to a particular circuit of neurons that produce the neurotransmitter oxytocin , which is known to affect social affiliation in many species . Investigation of other neurotransmitters revealed that these particular neurons also produce corticotropin-releasing hormone , which is known to regulate the response to anxiety and stress . Wircer et al . found that orthopedia regulates how much of each neurotransmitter is coproduced by the same neurons . This ability to change the balance of neurotransmitter production may allow the fish to switch between the “social” and “stress” states , enabling them to rapidly adapt to environmental changes and change their behavior . Exactly how orthopedia regulates the balance of neuropeptide production – and how this influences behavior – remains a question to be answered by further studies . More work is also needed to determine how these results relate to what occurs in the brains of mammals.

Figure 7: Ground-truth sample summary.

between the two methods. However, we will include the code for the BertSum version and the dataset in our submission for your perusal.

Despite operating under the constraints of limited resources and datasets, our fine-tuned model demonstrated its capacity to generate satisfying abstracts from extensive scientific papers in the biological domain. It outperformed the baseline model across all evaluation metrics, including ROUGE, BertScore, and METEOR. Notably, its readability and accessibility were on par with the advanced GPT-3.5 method and human-authored ground truth.

We recognize that our model occasionally produces repetitive content. To address this, we suggest the acquisition of more data and further fine-tuning of the model. The introduction of a repetition penalty during testing could further enhance the model's performance.

The significance of our project is underscored by its ambitious aim: to create concise summaries of extensive documents within a resource-limited environment. This endeavor has led to considerable progress in the field, showcasing the potential of our approach. Our approach can definitely be expanded upon with further work and research.

9.2 Future work

If we were to continue working on our project in the future, we would try to get PreSumm working. We would try to use PreSumm's extractive model to get extractive summaries that can then be fed into PEGASUS (abstractive). In order to be able to do that, we would need to switch to a different dataset that has both extractive and abstractive ground-truths. Another thing we would try is to hyperparameter tune BERTScore further and better figure out which version of BERT to use for it.

10 AI Disclosure

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used.
  - Yes, ChatGPT

If you answered yes to the above question, please complete the following as well:

1. Give me some example a human evaluation guideline for abstractive summary
2. Given a sentence, please replace the uncommon words with the most likely synonyms or hypernyms
3. Can you help me to polish the following paragraph: ”some paragraph from writing”
4. How do I include JSON format in the latex file?
5. I want to rank the sentences in the passage, is sentence Bert a good method for that?
6. Please give me a format that I can use for README.md
7. what's PEGASUS-X? relationship from PEGASUS
8. Use numpy to pick 100 random indices from a numpy array

9. Which is better for summarization, METEOR or COMET

- **Free response:** For each section or paragraph for which you used assistance, describe your overall experience with the AI. How helpful was it? Did it just directly give you a good output, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to generate new text, check your own ideas, or rewrite text?
  1. It was helpful. It generated a guideline that we were able to build off of.
  2. It was helpful. It replaced words with synonyms and hypernyms.
  3. It was helpful. It polished the paragraph and I tweaked it a bit afterwards.
  4. It was helpful. It gave me the exact code I needed.
  5. It was helpful. It supplemented what I got from lectures and online.
  6. It was helpful. It gave me the format as I needed it.
  7. It was helpful. It supplemented other information from the internet.
  8. It was helpful. It directly gave me the answer. It wasn't wrong. I used it to generate the code needed for getting 100 random numbers.
  9. It was helpful. It supplemented what I already knew and other sources like the lectures. It seemed like a good answer and it made sense

References

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding.

Du, Y., Li, Q., Wang, L., and He, Y. (2020). Biomedical-domain pre-trained language model for extractive summarization. *Knowledge-Based Systems*, 199:105964.

Goldsack, T., Zhang, Z., Lin, C., and Scarton, C. (2022). Making science simple: Corpora for the lay summarisation of scientific literature.

Karpinska, M. and Iyyer, M. (2023). Large language models effectively leverage document-level context for literary translation, but critical errors persist.

Kim, S. (2020). Using pre-trained transformer for better lay summarization. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 328–335, Online. Association for Computational Linguistics.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Liu, Y. (2019). Fine-tune bert for extractive summarization.

Liu, Y. and Lapata, M. (2019). Text summarization with pre-trained encoders.

Phang, J., Zhao, Y., and Liu, P. J. (2022). Investigating efficiently extending transformers for long input summarization.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks.

University, P. (2010). About wordnet. http://wordnet.princeton.edu.

Yang, Z., Zhu, C., Gmyr, R., Zeng, M., Huang, X., and Darve, E. (2020). Ted: A pretrained unsupervised summarization model with theme modeling and denoising.

Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2021). Big bird: Transformers for longer sequences.

Zhang, J., Zhao, Y., Saleh, M., and Liu, P. J. (2020). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization.