Master's Thesis

Accelerating Systematic Reviews: Developing a Pipeline for Creating Enhanced Abstracts

Kevin Patyk (8623659)

Supervisors: Ayoub Bagheri & Rens van de Schoot

Methodology and Statistics for the Behavioral, Biomedical, and Social Sciences

Utrecht University

May 9, 2022

Candidate Journal: *IEEE Xplore*

Abstract

Systematic reviews are valuable because they summarize what is and is not known on a specific topic using a rigorous and transparent methodology. However, the screening process of systematic reviews is time-consuming and prone to human error. Recent developments in machine learning have sought to facilitate the screening process through the use of automated technologies. One such program is ASReview, which uses reinforcement learning to reduce the number of articles that need to be screened manually. Although ASReview performs well in previous studies, it is only able to present the abstract to the user, which may not provide enough information to make a decision about inclusion status. The goal of this thesis is to establish a pipeline for converting PDF documents to a clean text format, which can then be used to automatically make summaries of the full text (enhanced abstracts). In total, 15 text summarization algorithms are tested and evaluated using an open test database. Then, the best performing and most practical algorithm is used to generate summaries of available full texts used for a meta-analysis on depression. Finally, a simulation study is conducted to determine how much time the automated summaries save during the screening process in comparison to the original abstracts and full text. The results show that the pipeline is successful in converting PDFs into a clean text format which can be used to make enhanced abstracts for use in systematic reviews. The simulation study demonstrates that the enhanced abstracts performed marginally worse relative to the original abstracts and full text, but still save a significant amount of time compared to manual screening. Follow-up research is needed to draw more concrete conclusions about the performance of the enhanced abstracts. Areas of improvement and suggestions for future research are provided.

Keywords: *automated systematic reviews, text summarization, machine learning, screening prioritization, text mining*.

Introduction

A systematic review is a method that involves finding all potentially relevant studies on a specific phenomenon to summarize and draw conclusions about what is and is not known in a thorough manner [1]. The systematic review methodology is valuable because it follows a fixed process for all reviews, thus allowing for an objective, rigorous, and transparent approach which is unbiased and facilitates replicability [2]. This methodology is crucial for researchers to summarize the state of affairs in a specific field. Moreover, it can help inform public policy and practice, and provide valuable information to clinicians [3-4]. However, the systematic review method has drawbacks, specifically during the screening process. In the guidelines outlined by Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA), potential eligible articles are screened by reviewers by first checking titles, then abstracts, and finally full text to decide about inclusion or exclusion [5]. Not only is this screening process extremely time-consuming, but it is susceptible to human-induced error and fatigue [2] [6]. For example, Wang et al. (2020) demonstrated that, during abstract screening, the total error rate (false inclusion or false exclusion) for human reviewers is 10.76% [7]. The screening process in systematic reviews has always been challenging, but it is further exacerbated by the influx of new databases to search and the ever-increasing quantity of published papers and journals [6].

A recently developed, open-source tool that accelerates the screening process through active learning is ASReview [8]. Active learning is a specific type of machine learning algorithm that interacts with the user and can select the data it uses for learning [9]. In the context of systematic reviews, active learning can reduce the number of records that need to be screened manually [10]. The pipeline for using ASReview starts with records identified via a systematic search. The user needs to label a minimum set of records with basic article information, herein

referred to as metadata, such as titles, abstracts, journals, and authors, to train the first iteration of the model. In what is called the active learning cycle, the machine learning model selects a record to present to the user based on the relevance score it predicted using article titles and abstracts. The user then screens the record and provides a label to determine if it is relevant or irrelevant to their systematic review; the model is then retrained using this information. This cycle continues until the user stops labeling. In the last step of the process, the user receives a set of relevant records to be used in their systematic review. Simulation studies have demonstrated that active learning provides a high quality and efficient alternative to manual text screening [11-13].

One drawback to the implementation of active learning in ASReview is that the software only presents the abstract as it is available in the metadata. The user can search for the full text outside the system to decide on relevance, but this information cannot be used by the machine. Active learning is not specific to abstracts; it can use the full text, but this would be computationally expensive and add noise to the system in the form of unnecessary textual information, which can lower performance. On the other hand, abstracts alone often do not always provide enough information to decide whether a study is relevant or irrelevant to the research question. A quick solution is for the user to read the full text outside the system, decide on relevance, and add a label to the system. However, this strategy is not only inconvenient and time-consuming, but also the properties of the full text are unknown to the active learning model. This process, herein referred to as algorithm-out-of-loop active learning (AOL-AL), can result in the active learning model not knowing that certain properties of the text are important and that they should be used as a classifier when deciding on relevance. Furthermore, AOL-AL may be susceptible to human error and fatigue because the reviewer must read the full text and make a decision, similar to a traditional systematic review. If the active learning algorithm learns from

errors made by human reviewers, the model will become unreliable, and the quality of the results will deteriorate [14]. A solution to this issue is using text summarization, a cornerstone method within the field of text mining, which has so far not yet been applied to ASReview.

Text mining is a process that uses algorithms to discover previously unknown information and non-trivial patterns from unstructured or semi-structured text documents [15-16]. Within the field of text mining, many techniques exist, such as text retrieval, text categorization, and text summarization, that facilitate identifying relevant literature, categorizing it, and summarizing it [17-18]. Text summarization started in 1958 and has since become a key method in the field because it can produce a concise summary of long documents while preserving the most essential information [19-20]. According to Huang et al., the four primary goals of text summarization are text coherence, redundancy, significance, and information coverage [21].

The emergence of the internet, online publishing, and the development of an electronic government have created a larger need for text summarization [22]. Individuals are now finding it difficult to find relevant information because of the massive number of online documents and texts [22]. This has caused exponential growth in the field of text summarization, since more advanced and streamlined methods are needed to summarize the vast amounts of text available electronically [23]. Text summarization can prevent readers from becoming exhausted while reading large amounts of text, resulting in them omitting essential information [22]. In recent years, researchers have attempted to find ways to improve text summarization through applying more complex mathematical and computational methods, such as conditional Markov models and statistical language models [18]. However, text summarization can be a complex and challenging process because computers do not have human knowledge and the capability to understand human language [20]. Furthermore, many factors, such as sentence ordering, redundancy, document

format, and compression ratio must be taken into account [24-25]. Finally, elements such as language (i.e., Dutch versus German), document type (single document versus multi-document), and document length must be considered when developing text summarization methods [26].

Machine learning and deep learning methods are powerful techniques that can expedite and improve text summarization [23]. In recent years, many advanced methods for text summarization have been developed, which focus on implementing machine learning-based methods and optimization techniques, such as Hidden Unit Bidirectional Representations from Transformers (HuBERT) and multilingual text-to-text transformer (mT5) [27-28]. Text summarization methods that incorporate machine learning-based approaches have outperformed commercial text summarizers and can summarize text with a high degree of accuracy [29-30]. These machine-learning based text summarization approaches can be applied to various types of documents, such as medical documents, legal documents, and published scientific articles, to reduce the amount of time and effort it takes to read and screen them [31-32].

In this thesis, we investigate the creation of a pipeline for identifying, downloading, parsing and preprocessing the full text of PDF articles into a clean format, combining the full text with metadata, and creating enhanced abstracts. In what follows, we first test and evaluate 15 separate machine-learning based text summarization algorithms to determine which is most accurate based on similarity metrics between an algorithm-generated summary and a gold-standard human summary. After selecting the best performing algorithm, while also considering applicability (i.e., max input length, max output length, on what sort of data it is pre-trained on), it is used to create summaries of research articles and add them to an already existing dataset containing article metadata (titles, abstracts, etc.), but not summaries. Finally, a simulation study is conducted using ASReview to determine whether the enhanced abstracts outperform the original abstracts and the

full text using two performance metrics (average time to discovery and work saved over sampling). This study has been approved by the Utrecht University Faculty Ethics Assessment Committee (FETC). All scripts to reproduce the results can be found on GitHub (https://github.com/Kevin-Patyk/ASReview-Text-Summarization-Thesis).

Methods

The study is broken down into three separate stages (Algorithm Evaluation and Selection, Preprocessing and Algorithm Application, and Simulation Study), outlined in detail below. Python (version 3.10) is used to conduct all analyses in the first stage (i.e., application and evaluation of the specified machine learning-based algorithms) and to later create the enhanced abstracts in the final stage, R (version 4.2.0) is used for preprocessing, and ASReview LAB (version 0.19) is used to conduct the simulation study in the last stage. The data used in the study is openly available and is described in detail in the following section.

Data

For this study, two datasets are used. In the first stage (Algorithm Evaluation and Selection), the PubMed dataset for summarization collected by Cohan et al. (2018) is used [33]. The dataset contains paper identification numbers, the full text from the scientific articles, and the abstracts corresponding to each article. There are a total of 133,215 articles that are split into training, validation, and test sets. The training set consists of 119,924 articles, the validation set consists of 6,633 articles, and the test set consists of 6,658 articles. The average document length is 3,016 words and the average length of the associated abstracts is 202 words. This data is publicly available and can be found on the collecting author's GitHub repository (https://github.com/armancohan/long-summarization).

In the second and third stages (Algorithm Application and Simulation Study, respectively),

the systematic review data from “Psychological theories of depressive relapse and recurrence”, herein called the depression data, collected by Brouwer et al. (2019) is used [34]. This data contains 46,376 records with metadata (i.e., titles and abstracts) and labeling decisions (63 articles are labeled relevant, the rest as irrelevant). The data is available on the Open Science Framework (OSF) (https://osf.io/r45yz/).

Stage 1: Algorithm Evaluation and Selection

The first stage of research involves finding and testing a text summarization algorithm. Four previously-developed machine learning-based text summarization methods are used. These include Bidirectional and Auto-Regressive Transformers (BART), T5, Pre-trained with Extracted Gap-sentences for Abstractive Summarization (PEGASUS), and BigBird PEGASUS [35-38]. Along with applying and evaluating the base models (with no pre-training), models pre-trained on different data sets are also included. In total, 15 models are evaluated and uploaded to Hugging Face (https://huggingface.co/Kevincp560). Additionally, the scripts to run the models can be found on GitHub (https://github.com/Kevin-Patyk/ASReview-Text-Summarization-Thesis). All of these algorithms have been described in-depth in their respective publications and the technical details are not discussed here; however, brief summaries of each of them can be found in Appendix A. To reduce training time, the BART and T5 models are trained on a reduced version of the PubMed dataset with 8,000 articles in the training set and 2,000 in the validation set. Furthermore, to train PEGASUS and BigBird-PEGASUS, the data is further reduced to 2,000 articles in the training set and 500 in the validation set; this is because more advanced graphics processing units (GPUs) are needed and there is limited availability of such resources for the author. All efforts made by the author to access more resources (stronger GPUs) to run all of the models accordingly are described

in Appendix B. Furthermore, a full list of model specifications can be seen in Table 1 of Appendix C.

The accuracy of the algorithms are evaluated using Recall-Oriented Understudy for Gisting Evaluation (ROUGE) metrics developed by Lin (2004) [39]. The ROUGE metrics measure the similarity between a candidate summary, which is produced using a text summarization algorithm, and a human-generated reference summary, which is provided by the abstracts in the PubMed dataset. The ROUGE metrics compare the summaries and calculate a score between 0 and 100 on four metrics:

- ROUGE-1: refers to the candidate's and reference summaries' overlap of unigrams.
- ROUGE-2: refers to the candidate's and reference summaries' overlap of bigrams.
- ROUGE-L: computes the longest common subsequence (LCS) between a candidate and reference summary while ignoring new lines of text.
- ROUGE-L Summary: The LCS is computed between each pair of candidate and reference sentences while including new lines of text as sentence boundaries.

One of the models is chosen based on how well it performs relative to the other algorithms using the above metrics, the amount of input tokens that the model can handle, how long of a summary it can generate, and on what sort of data the pre-training occurred on. This stage is crucial in determining how accurate the model-generated summaries are when compared to a gold standard summary and the applicability of the outlined text summarization methods, as the best-performing and most applicable method is applied in the second stage.

Stage 2: Preprocessing and Algorithm Application

In the second stage of this research, the depression dataset is used. The depression dataset already contains titles, abstracts, and labeling decisions. The first goal of this stage is to identify

all of the open access articles to avoid copyright issues. Once the open access articles are identified, articles that have a PDF as the URL are located and downloaded. After downloading all of the articles, the objective is to parse them (extract text) into XML format so that they can be imported into R. The scripts used for identifying which articles have a PDF as the URL and importing the XML files into R are provided by Bianca Kramer (https://www.uu.nl/medewerkers/bmrkramer). The parsing of the PDF files is done using GROBID (https://github.com/kermitt2/grobid) [40]. The second goal of this stage is to perform pre-processing, which consists of removing any articles containing non-English text, having missing titles, or incomplete or incorrect PDFs. Furthermore, all text must be made lowercase and have punctation and non-UTF-8 characters removed so that only alpha-numeric characters remain.

After preprocessing, the full text is merged with the metadata and is exported into three separate datasets. The first dataset consists of only article metadata with original abstracts and no full text, while the other two datasets contain the metadata and the full text in place of the original abstracts. The reason for having an extra dataset with full text is because one will be used to process the full text into enhanced abstracts using the best performing and most applicable method from Stage 1. The final output from this stage consists of three datasets, one with only article metadata and original abstracts, one with article metadata and full text in place of the original abstracts, and one with article metadata and enhanced abstracts in place of the original abstracts. The reason for replacing the abstracts with full text and enhanced abstracts in two of the datasets is because the input for the ASReview simulation study in the next stage requires only one column to serve as an abstract and contain the text which will train the model.

Stage 3: Simulation Study

In the third and final stage, a simulation studying using ASReview (version 0.19) is conducted. The ASReview simulation mode allows the user to run simulations on previous, fully-labeled datasets to determine how much time can be saved using different ASReview active learning models (as opposed to manual screening). The simulation replicates the screening process by iterating through the dataset (like an ASReview user would typically do) using information regarding inclusion or exclusion to learn during the active learning cycle. The simulation is run using the default settings of ASReview simulation mode: naive bayes as the classification method, max as the query strategy, double as the balance strategy, and term frequency-inverse document frequency (TF-IDF) as the feature extraction method. The simulation is set up so, that for each relevant record, one run is performed with 10 randomly picked irrelevant records that remain constant across runs. When there are several datasets, the simulation will iterate through all datasets and run one simulation per inclusion. In this case, a total of 12 simulation are performed.

The three datasets (original abstracts vs. full text vs. enhanced abstracts) are compared on two metrics which are previously used by van de Schoot et al. [12]. These include:

- Average Time to Discovery: How long it takes, on average, to find a relevant article.
- Work Saved Over Sampling at 95% Recall: A measure that represents how much work the reviewer saved by using active learning rather than manual screening, assuming 5% of the relevant publications are not identified.

These metrics determine if the performance of active learning is better on the original abstracts, full text, or enhanced abstracts.

Results

Stage 1: Algorithm Evaluation and Selection

When tested on the PubMed data, BigBird PEGASUS pre-trained on ARXIV performs the best (ROUGE-1 = 45.58, ROUGE-2 = 20.02, ROUGE-L = 28.36, ROUGE-L Sum = 41.46), followed by BigBird PEGASUS pre-trained on Big Patent (ROUGE-1 = 45.09, ROUGE-2 = 19.55, ROUGE-L = 27.39, ROUGE-L Sum = 41.11), and then by PEGASUS pre-trained on ARXIV (ROUGE-1 = 44.29, ROUGE-2 = 19.05, ROUGE-L = 27.11, ROUGE-L Sum = 40.26). A detailed list of the evaluation results for all models evaluated can be found in Table 2 of Appendix D. Furthermore, a visualization of algorithm performance can be seen in Figures 1-4. BigBird PEGASUS pre-trained on ARXIV data is selected to generate the enhanced abstracts in the second stage. This is for several reasons. Firstly, it is because, as stated before, BigBird-PEGASUS pre-trained on ARXIV data outperforms all the other models in terms of the evaluation metrics. Secondly, BigBird PEGASUS is capable of handling a maximum input length 4,096 tokens. This is four times more than PEGASUS and BART, which both have a maximum input length of 1,024, and eight times more than T5, which has a maximum input length of 512. This is crucial because scientific articles are classified as long documents with a large amount of input. In particular, the average document length (in words) for the PubMed data from the first stage is 3,016, with some articles extending beyond 4,096. Upon reaching the maximum number of tokens the algorithm can handle, truncation occurs and any additional tokens are dropped. Thus, if an algorithm cannot handle a large amount of input, the resulting summary will be based on incomplete information. Furthermore, BigBird PEGASUS is capable of generating longer summaries, ranging from 219 to 231 words. This is more than BART (19 to 142 words) and T5 (19 words), although similar to PEGASUS (125 to 231 words). This is important because, with

such a large input length, the corresponding algorithm-generated summary should be long enough to contain the most important information. Lastly, BigBird PEGASUS pre-trained on ARXIV is chosen because the pre-training occurred on scientific articles, which is more appropriate to later create summaries of scientific articles than algorithms pre-trained on unrelated material.

Figure 1

ROUGE-1 Scores for All Models after 5 Epochs

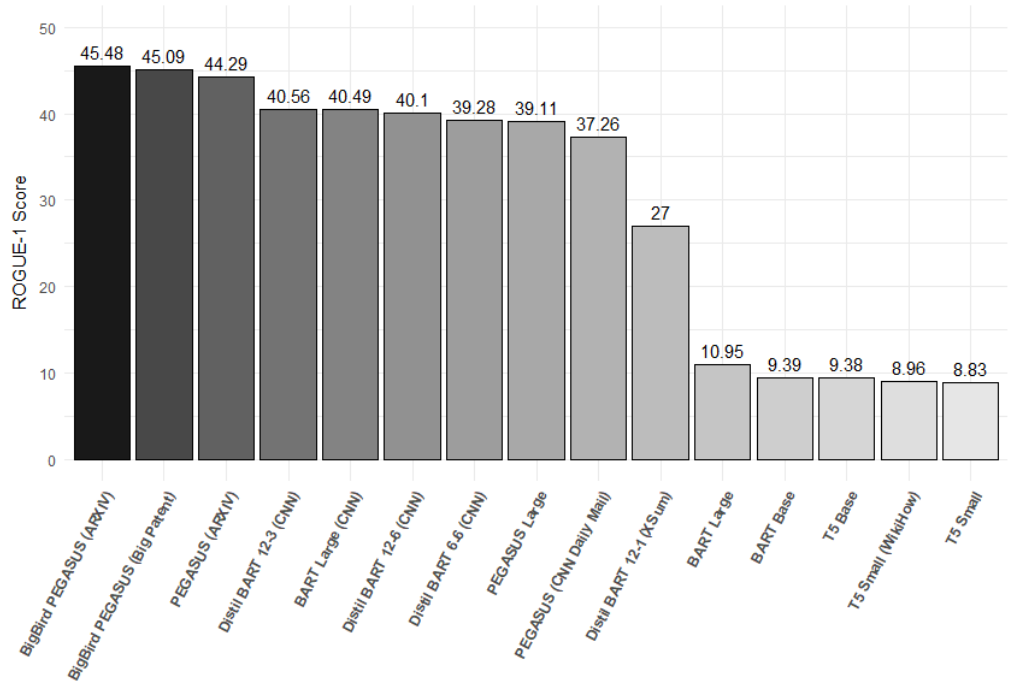


Figure 2

ROUGE-2 Scores for All Models after 5 Epochs

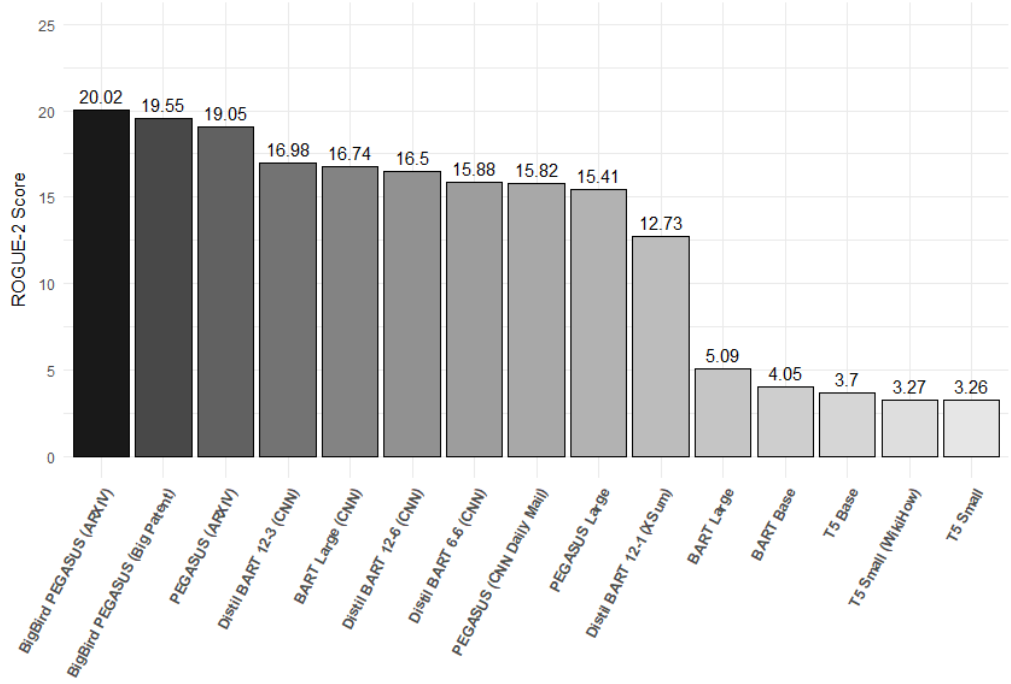


Figure 3

ROUGE-L Scores for All Models after 5 Epochs

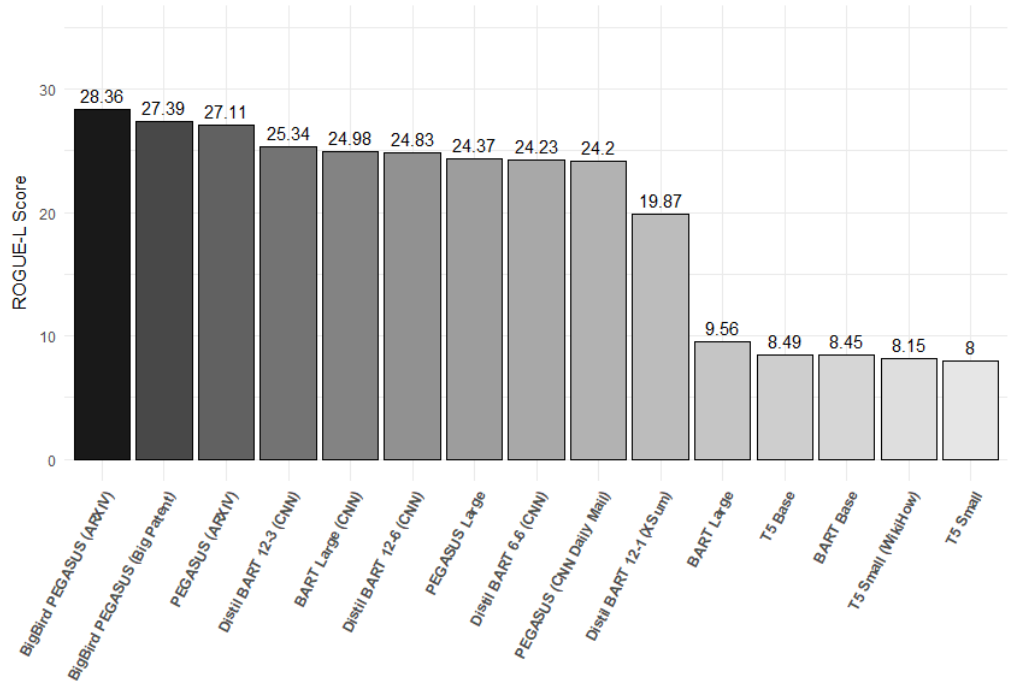
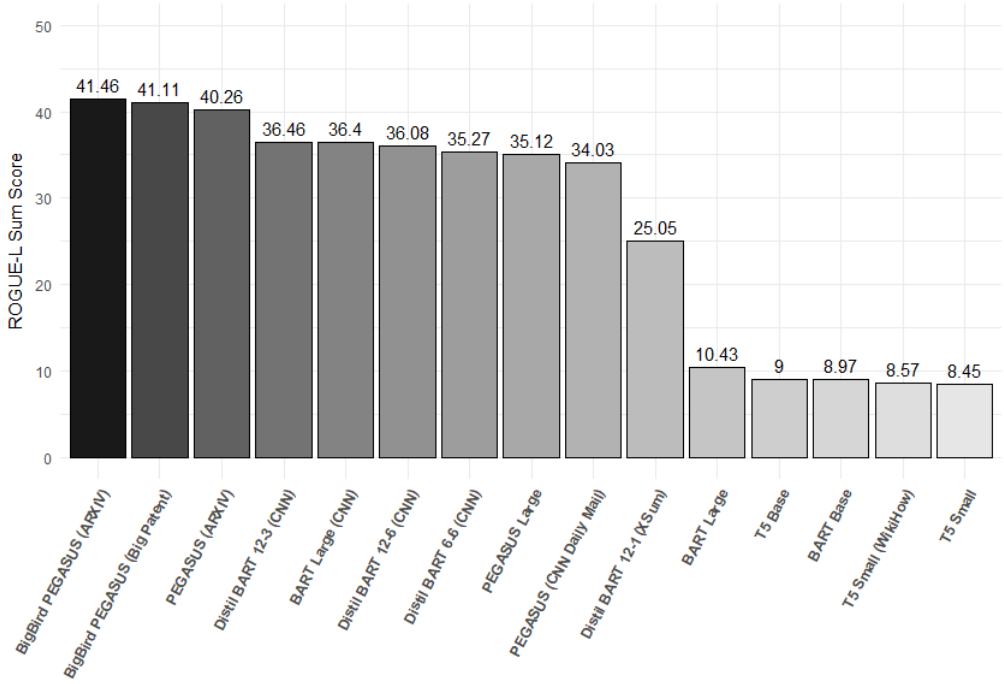


Figure 4

ROUGE-L Summary Scores for All Models after 5 Epochs

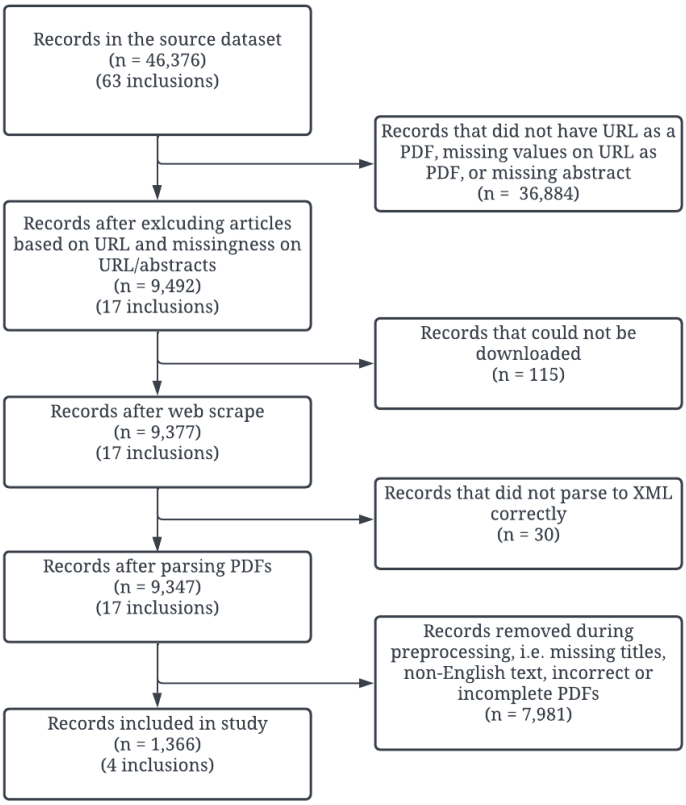


Stage 2: Preprocessing and Algorithm Application

Originally, there are 46,376 articles in the dataset. After removing records that did not have the PDF as the URL, had missing values on the URL, or did not have an abstract, 9,492 articles remained. 115 of those records are not properly downloaded during the web scrape, leaving 9,377 articles. 30 of the articles did not parse correctly into XML format and are dropped as a result, leaving the total at 9,347. Finally, during preprocessing, 7,981 are removed for missing titles (abstracts are being matched with full text based on titles), having non-English text (the algorithm is not multilingual and is designed for English text), or having incomplete or incorrect PDFs. After preprocessing, 1,366 articles with four inclusions remained. A flow diagram depicting reasons for article removal can be seen in Figure 5.

Figure 5

Flow Diagram Depicting Reasons for Article Removal During Preprocessing



Stage 3: Simulation Study

In the simulation study, for every dataset, four runs are performed with one inclusion and 10 random exclusions, which are kept constant across the runs. In total, there are 12 runs. For the dataset containing the enhanced abstracts in place of the original abstracts, the average work saved over sampling at 95% recall is 85% and ranges from 78% to 91%. Thus, 95% of the relevant studies will be found after screening between 9% to 22% of the articles. The mean average time to discovery for this dataset is 133. Hence, it takes an average of 133 article screenings before a relevant one is found. For the dataset containing the original abstracts, the average work saved over sampling at 95% recall is 94% and ranges from 93% to 94%. Thus, 95% of the relevant studies will be found after screening between 6% to 7% of the articles. The mean average time to discovery

for this dataset is 46. Thus, it takes an average of 46 article screenings before a relevant one is found. For the dataset containing the full text in place of the original abstracts, the average work saved over sampling at 95% recall is 91% and ranges from 89% to 94%. Thus, 95% of the relevant studies will be found after screening between 6% to 11% of the articles. The mean average time to discovery for this dataset is 68. Hence, it takes an average of 68 article screenings before a relevant one is found. A list of the results from the simulation can be found in Table 3.

Although the enhanced abstracts performed worse compared to the original abstracts and full text, there is still a major improvement over manual screening and results in a large increase in time savings. Thus, the pipeline achieves the goal of identifying, downloading, parsing and preprocessing the full text of PDF articles, which can then be used to make enhanced abstracts that greatly facilitate the screening process. Moreover, the difference between the performance of the enhanced abstracts compared to the original abstracts is about 10%. Lastly, the difference between the performance of the original abstracts and the full text is the variance in performance.

Table 3
Simulation Results for All Variations of Text Used (Enhanced Abstracts vs. Original Abstracts vs. Full Text)

Model Name	Average WSS[3]	Lower Bound WSS[4]	Upper Bound WSS[5]	Mean ATD[6]
Enhanced Abstracts	85%	78%	91%	133
Original Abstracts	94%	93%	94%	46
Full Text	91%	89%	94%	68

[1] For every inclusion (four) per dataset (three) one simulation is run, resulting in a total of 12 simulations.
[2] Each simulation is run using the default parameters of ASReview simulation mode. These settings include naive bayes as the classification method, max as the query strategy, double as the balance strategy, TF-IDF as the feature extraction method, and using 10 exclusions for every one inclusion.
[3] Average work saved over sampling at 95% recall (WSS) is the mean WSS across all four runs.
[4] The lower bound WSS is the lowest WSS of all four runs.
[5] The upper bound WSS is the highest WSS of all four runs.
[6] The mean average time to discovery (ATD) is the four individual ATDs summed and divided by the number of runs.

Discussion

A pipeline is designed that would facilitate converting a full text article into an algorithm generated summary (enhanced abstract). The enhanced abstract could then be used in place of an original abstract if the original abstract did not have enough information to make a decision about inclusion status during the screening process when using ASReview. The pipeline is successful in identifying, downloading, extracting, and preprocessing the full text from a PDF, which can then be used to create enhanced abstracts. This pipeline, to the author's knowledge, is one-of-a-kind and is able to reduce the workload of converting the full text in the PDF of a scientific article into summary. The ability to quickly generate the summary of an article is highly beneficial when the original abstract does not provide enough information to make a decision about inclusion status

when using ASReview. The ability of ASReview to provide an enhanced abstract not only saves time, but also saves the user the inconvenience of reading the full text outside of the system and manually adding a label. Along with the inconvenience factor, reading the full text and manually adding a label results in the active learning model not knowing the properties of the full text, since it is not available in the software. AOL-AL will result in the active learning model not knowing which properties to use as classifiers when deciding on relevance and is prone to human-error. Being able to provide an enhanced abstract and prevent these issues can make systematic reviewing using ASReview more streamlined, efficient, and flexible.

The simulation study conducted after the development of the pipeline using the three datasets (original abstracts vs. full text vs. enhanced abstracts) demonstrated that the enhanced abstracts performed worse relative to the others. The enhanced abstracts had the lowest work saved over sampling and the highest average time to discovery. This indicates that it would take more screening time to find relevant articles when using the enhanced abstracts compared to the original abstracts or full text for this round of simulations, but not by a large margin. The difference between the enhanced abstracts compared to the original abstracts and full text is about 10%. This indicates that the enhanced abstracts still save an enormous amount of time compared to manual screening. One possible explanation for such results is the small number of inclusions (four) that are used in the simulation study. A large portion of the included papers (13) are removed during the downloading, converting, and preprocessing stage for various reasons, as shown by Figure 5. Having a small sample size of included papers during the simulation study can deteriorate the quality of the results. Furthermore, the simulation study is conducted using the default settings of the ASReview simulation mode. ASReview simulation mode has a plethora of options for how the simulation study can be conducted, and the present study could not explore all of these options

and determine their performance. Thus, any future editions or follow-up studies can focus on increasing the number of inclusions and exploring the various settings for the simulation.

In addition to the limitations present during the simulation study, the developed pipeline also contains some drawbacks. First, the pipeline can be computationally expensive. If an author is starting with only the links to the PDF files, then downloading thousands of PDFs, parsing, importing, and preprocessing them takes a large amount of time. Additionally, BigBird PEGASUS is a vastly complex deep-learning algorithm and can take a long time to generate summaries of long scientific articles. Generating a summary using BigBird PEGASUS can be accelerated by optimizing it through the use of a GPU. However, one needs access to a powerful enough NVIDIA GPU to achieve this optimization, which can be difficult to access. The reason for needing access to an NVIDIA GPU specifically is because the software for optimizing models using a GPU (CUDA) is developed by NVIDIA. Second, parsing PDFs is a difficult process and techniques to do so efficiently and quickly are still being optimized. Although the machine-learning based GROBID is used, parsing a PDF can result in a lot of text coming from figures or tables that may or may not be wanted. Moreover, the formatting of tables or figures can be leftover after being parsed (i.e., tables with cells will be converted into text (“<cell>”) and so will figures (“<figure>”)). Lastly, GROBID converts the parsed text into XML files which can be cumbersome to work with because they are split into many sections, such as title, body (main text), references, etc., which can be difficult to extract and load into programming languages.

Third, when cleaning the text during preprocessing (i.e., converting to lower case, removing non-alpha-numeric characters, removing punctation), some important information may be lost because of the varying structure of the articles. For example, during this study, anything between parentheses in the text is removed so that in-text citations would not provide unnecessary

information to the algorithm and take up the limited amount of tokens which can be input. However, not all publishers follow the same style for in-text citations and important information can be contained within parentheses, which will then have been removed. Additionally, there can be some important information represented by non-alpha-numeric characters, such as authors reporting statistics (i.e., $p > 0.05$, $r = 0.35$). Removal of these non-alpha-numeric characters can result in incomplete or confusing information being generated by the text summarization algorithm (i.e., multiple "p" or "r" characters or random numbers not attached to anything scattered throughout the text and resulting summary since the symbols ">" and "=" will be removed). Lastly, the algorithm used to generate summaries is pre-trained strictly on articles from the publisher ARXIV, which focuses on science, technology, engineering, and mathematics (STEM). It is then later applied to articles related to depression. Using an algorithm to generate summaries that is pre-trained on unrelated material may reduce accuracy.

Future research or follow-up studies should focus on improvements to the downloading, parsing, and preprocessing steps in the pipeline. One suggestion for improvement is working with publishers to get access more easily to the articles. This has several benefits. The first is that it would ensure that the articles being downloaded are, in fact, the correct articles, are in the proper format, and can be easily downloaded. During preprocessing, 7,891 out of 9,347 (84%) articles are removed due to being incomplete or the incorrect PDF having been downloaded, drastically reducing the sample size. Moreover, 36,884 out of 46,376 (80%) are removed during preprocessing, a majority of them for not having the URL as a PDF, thus not being easily accessible via web scrape. If publishers provided a way to easily access and download the requested articles, these issues could be avoided. The second benefit is that the preprocessing stage would become more streamlined and accurate. Instead of having to make general assumptions about how the text

should be cleaned, preprocessing could be grouped by publisher, and publisher-specific regular expressions or text cleaning pipelines could be developed and used to remove unwanted text. This way, possibly important information is not lost during text cleaning. Lastly, publishers may be able to offer the articles in a different format than PDF, such as a LaTeX source file. This would significantly reduce the workload during preprocessing because the PDFs would no longer need to be downloaded, parsed, and have the text extracted. Moreover, this would make the text much easier to clean. These steps can be long and computationally expensive, and having access to the text immediately would circumvent the need to perform these time-consuming steps.

Once these limitations have been addressed, the end goal should be to use the pipeline to curate a massive dataset (500,000 – 1,000,000) of scientific articles on many different topics and from a variety of publishers and journals. Then, a text summarization algorithm can be trained on this dataset to create a general model which can be implemented into ASReview to aid systematic reviewing. Providing an option to create a summary of an article when the abstract does not provide enough information about inclusion status for authors conducting systematic reviews would further reduce screening time, effort, and prevent AOL-AL. A systematic reviewing dataset and general systematic reviewing text summarization model could be made publicly available (for example, on Hugging Face) for other individuals in the scientific community to use and experiment with. In addition to providing accessibility to such a dataset and model, this could serve as a major step for advancing automated systematic reviewing and machine learning-based summarization of scientific literature.

To conclude, a pipeline for identifying, downloading, parsing, extracting, and summarizing the text from the PDF of a scientific article is successfully developed. Although the simulation study demonstrated that the enhanced abstracts performed worse compared to the original abstracts

and full text, the difference is marginal and using the enhanced abstracts still results in significant time savings over manual screening. Additionally, the information found during the development of the pipeline is invaluable to the future of automated systematic reviews using ASReview. Being the first of its kind, to the best of the author's knowledge, this pipeline provides us with directions for future research and where improvements can be made to further enhance and streamline the process. With the application of such knowledge, providing an accurate and quick summary of a full text article during the automated screening process of a systematic review using ASReview is certainly feasible.

Acknowledgements

This work would not have been possible without the help and support of my amazing supervisors, Ayoub Bagheri and Rens van de Schoot. Thank you for always being supportive, available, and incredibly helpful throughout the course of this study. I would also personally like to thank Bianca Kramer for providing me with invaluable assistance with identifying, downloading, and parsing the PDF files. Furthermore, a very special thank you to Jelle Teijema for helping me with the simulation study and identifying areas for improvement. Lastly, I would like to thank the entirety of the ASReview team for being supportive and curious about the work. I am very excited and interested to see how this work is incorporated into ASReview in the future.

References

1. Denyer, D., & Tranfield, D. (2009). Producing a systematic review.
2. Mallett, R., Hagen-Zanker, J., Slater, R., & Duvendack, M. (2012). The benefits and challenges of using systematic reviews in international development research. *Journal of development effectiveness*, 4(3), 445-455.
3. Gough, D., & Elbourne, D. (2002). Systematic research synthesis to inform policy, practice and democratic debate. *Social policy and society*, 1(3), 225-23.
4. Oliver, S., Dickson, K., & Bangpan, M. (2015). Systematic reviews: making them policy relevant. A briefing for policy makers and systematic reviewers. *EPPI-Centre, Social Science Research Unit, UCL Institute of Education, University College London, London.*
5. Moher, David, Larissa Shamseer, Mike Clarke, Davina Ghersi, Alessandro Liberati, Mark Petticrew, Paul Shekelle, and Lesley A. Stewart. "Preferred reporting items for systematic review and meta-analysis protocols (PRISMA-P) 2015 statement." *Systematic reviews* 4, no. 1 (2015): 1-9.
6. O'Mara-Eves, A., Thomas, J., McNaught, J., Miwa, M., & Ananiadou, S. (2015). Using text mining for study identification in systematic reviews: a systematic review of current approaches. *Systematic reviews*, 4(1), 1-22.
7. Wang, Z., Nayfeh, T., Tetzlaff, J., O'Blenis, P., & Murad, M. H. (2020). Error rates of human reviewers during abstract screening in systematic reviews. *PloS one*, 15(1), e0227742.
8. van de Schoot, R., De Bruin, J., Schram, R., Zahedi, P., De Boer, J., Weijdema, F., Kramer, B., Huijts, M., Hoogerwerf, M., Ferdinands, G., Harkema, A., Willemsen, J., Ma, Y., Fang, Q., Tummers, L., & Oberski, D. (2021). ASReview: Active learning for systematic reviews (v0.14.2). *Zenodo*. https://doi.org/10.5281/zenodo.4432652
9. Settles, B. (2009). Active learning literature survey.
10. Cohen, A. M., Hersh, W. R., Peterson, K., & Yen, P. Y. (2006). Reducing workload in systematic review preparation using automated citation classification. *Journal of the American Medical Informatics Association*, 13(2), 206-219.
11. Ferdinands, G., Schram, R. D., de Bruin, J., Bagheri, A., Oberski, D. L., Tummers, L., & van de Schoot, R. (2020). Active learning for screening prioritization in systematic reviews-A simulation study.
12. van de Schoot, R., De Bruin, J., Schram, R., Zahedi, P., De Boer, J., Weijdema, F., Kramer, B., Huijts, M., Hoogerwerf, M., Ferdinands, G., Harkema, A., Willemsen, J., Ma, Y., Fang, Q., Tummers, L., & Oberski, D. (2021). An open source machine learning framework for efficient and transparent systematic reviews. *Nature Machine Intelligence*, 3(2), 125-133.
13. Wallace, B. C., Trikalinos, T. A., Lau, J., Brodley, C., & Schmid, C. H. (2010). Semi-automated screening of biomedical citations for systematic reviews. *BMC Bioinformatics*, 11(1), 1-11.
14. Sanders, H., & Saxe, J. (2017). Garbage in, garbage out: How purport-edly great ML models can be screwed up by bad data. *Proceedings of Blackhat 2017*.
15. Hearst, M. (2003). What is text mining. *SIMS, UC Berkeley*, 5.

16. Tan, A. H. (1999, April). Text mining: The state of the art and the challenges. *In Proceedings of the PAKDD 1999 Workshop on Knowledge Discovery from Advanced Databases* (Vol. 8, pp. 65-70). sn.
17. Thomas, J., McNaught, J., & Ananiadou, S. (2011). Applications of text mining within systematic reviews. *Research synthesis methods*, 2(1), 1-14.
18. Dang, S., & Ahmad, P. H. (2015). A review of text mining techniques associated with various application areas. *International Journal of Science and Research*, 4(2), 2461-2466.
19. Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2), 159-165.
20. Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). Text summarization techniques: a brief survey. *arXiv preprint arXiv:1707.02268*.
21. Huang, L., He, Y., Wei, F., & Li, W. (2010, April). Modeling document summarization as multi-objective optimization. *In 2010 Third International Symposium on Intelligent Information Technology and Security Informatics* (pp. 382-386). IEEE.
22. Gambhir, M., & Gupta, V. (2017). Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1), 1-66.
23. Tas, O., & Kiyani, F. (2007). A survey automatic text summarization. *PressAcademia Procedia*, 5(1), 205-213.
24. Goldstein, J., Mittal, V. O., Carbonell, J. G., & Kantrowitz, M. (2000). Multi-document summarization by sentence extraction. *In NAACL-ANLP 2000 Workshop: Automatic Summarization.*
25. Hahn, U., & Mani, I. (2000). The challenges of automatic summarization. *Computer*, 33(11), 29-36.
26. Zajic, D. M., Dorr, B. J., & Lin, J. (2008). Single-document and multi-document summarization techniques for email threads using sentence compression. *Information Processing & Management*, 44(4), 1600-1610.
27. Hsu, W. N., Bolte, B., Tsai, Y. H. H., Lakhotia, K., Salakhutdinov, R., & Mohamed, A. (2021). HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units. *arXiv preprint arXiv:2106.07447*.
28. Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., ... & Raffel, C. (2020). mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.
29. Kaikhah, K. (2004). Text summarization using neural networks.
30. Kyoomarsi, F., Khosravi, H., Eslami, E., Dehkordy, P. K., & Tajoddin, A. (2008, May). Optimizing text summarization based on fuzzy logic. *In Seventh IEEE/ACIS International Conference on Computer and Information Science (ICIS 2008)* (pp. 347-352). IEEE.
31. Altmami, N. I., & Menai, M. E. B. (2020). Automatic summarization of scientific articles: A survey. *Journal of King Saud University-Computer and Information Sciences.*
32. El-Kassas, W. S., Salama, C. R., Rafea, A. A., & Mohamed, H. K. (2021). Automatic text summarization: A comprehensive survey. *Expert Systems with Applications*, 165, 113679.

33. Cohan, A., Dernoncourt, F., Kim, D. S., Bui, T., Kim, S., Chang, W., & Goharian, N. (2018). A discourse-aware attention model for abstractive summarization of long documents. *arXiv preprint arXiv:1804.05685*.
34. Brouwer, M. E., Williams, A. D., Kennis, M., Fu, Z., Klein, N. S., Cuijpers, P., & Bockting, C. L. (2019). Psychological theories of depressive relapse and recurrence: A systematic review and meta-analysis of prospective studies. *Clinical Psychology Review*, 74, 101773.
35. 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. *arXiv preprint arXiv:1910.13461*.
36. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.
37. Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020, November). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *In International Conference on Machine Learning* (pp. 11328-11339). PMLR.
38. Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., ... & Ahmed, A. (2020). Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33, 17283-17297.
39. Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. *In Text Summarization Branches Out* (pp. 74-81).
40. Lopez, P. (2009, September). GROBID: Combining automatic bibliographic data recognition and term extraction for scholarship publications. *In International conference on theory and practice of digital libraries* (pp. 473-474). Springer, Berlin, Heidelberg.

Appendix A: Summary of Applied Algorithms

Name	Algorithm Summary
BART	BART is a denoising autoencoder. BART uses standard sequence-to-sequence transformer-based architecture. BART's architecture is based on a typical sequence-to-sequence transformer. A random noising approach is used to train the algorithm by introducing noise to text. The model is then taught to reconstruct the original text by optimizing the cross-entropy (negative log likelihood) between the original document and the output of the decoder. A bidirectional encoder is applied to the corrupted text and a left-to-right autoregressive decoder is used.
T5	T5 is an encoder-decoder model that has been pre-trained on a multi-task mixture of unsupervised and supervised workloads, with each task transformed to text-to-text. Self-attention, optional encoder-decoder attention, and a feed-forward network are all included in each of the encoder and decoder's 12 blocks. Each block's feed-forward networks are made up of a dense layer, a ReLU nonlinearity layer, and another dense layer.
PEGASUS	PEGASUS is a self-supervised technique that uses gap sentences generation (GSG) to pre-train transformer-based encoder-decoder models on large text. The masked language model (MLM) is used by PEGASUS to remove or mask significant sentences from input documents. The important sentences are then formed as one output sequence from the leftover sentences in a technique similar to extractive summary generation.
BigBird PEGASUS	BigBird PEGASUS is a sparse-attention-based transformer that is capable of reducing a quadratic dependency to a linear dependency in terms of memory. Sparse attention, as presented, can manage sequences up to 8 times longer than what was previously conceivable with similar technology. Transformer-based models can be extended to longer sequences using BigBird. BigBird PEGASUS combines BigBird technology, thus allowing it to overcome the quadratic dependency while maintaining the features of full attention models, and the previously developed PEGASUS. Combined together, BigBird PEGASUS can manage longer sequences and context, thus improving performance on NLP tasks, such as summarization.

Appendix B: Footnote Describing the Author's Efforts

When optimizing deep-learning models, a computer's graphics processing unit (GPU) is used in favor of the computer's central processing unit (CPU). This is because the GPU is capable of making the calculations much more quickly than a CPU, thus reducing the training time and evaluation of the model by a substantial margin. However, gaining access to recent and more advanced GPUs can be difficult, especially without a budget, as all cloud computing services require payment. Typically, the more complex a model is, the more video RAM (VRAM) is needed in a GPU for optimization. Furthermore, the GPU must be compatible with NVIDIA's CUDA, which is the software that allows the use of the GPU for deep-learning optimization. The GPUs that are compatible with CUDA are all made by NVIDIA. The BART and T5 models were able to be run on Google Colab Pro ($10 a month), which gives access to GPUs depending on the availability (the user does not get to choose). The available GPUs are NVIDIA Tesla K80 GPU, which contains 12G of VRAM, the NVIDIA T4, which contains 16G of VRAM, and the NVIDIA Tesla P100, which also contains 16G of VRAM. However, the more sophisticated models, PEGASUS and BigBird PEGASUS required more VRAM (>24G), so the analysis could not be conducted on Google Colab.

A variety of outlets were used to try and solve this issue. The IT department was contacted and access to the Utrecht University supercomputer was granted, but the computer is not optimized for deep-learning and did not have adequate GPUs for such a task. Then, access was granted to the department of Methodology and Statistics' computer, but it only has a NVIDIA 1080 Ti, which has only 12G VRAM. When these options did not pan out, the author tried several cloud computing services, such as Gradient, Google Cloud API, Paperspace, and Leafcloud. All of these services require a payment of some sort, while some come with free credits. The author attempted to use

the free credits to get access to an adequate GPU, but the service required the user to request more resources, which typically gets denied to users not directly paying for the service (users trying to spend their free credits). After exhausting all of the resources, the author tried to train the most complex models (PEGASUS and BigBird PEGASUS), using a CPU for optimization. This did not work either because the machines did not have enough RAM, disk space, or the training time increased substantially (some estimates were for over 1,000 hours). Thus, such an option was not feasible. After another search on the internet, a cloud computing company by the name of DataCrunch was found that offered free credits and immediate access to high-end GPUs, without the need to request additional resources.

With a mixture of free credits and some of the author's own money, the most complex models could be trained, but not with the same dataset size as the simpler models (8,000 training and 2,000 validation), because it would be financially costly, as more credits would have to be purchased. Hence, the decision was made to reduce the training set to 2,000 and the validation set to 500 so the models could be applied and evaluated. Reducing the training and validations sets was the best option outside of paying for more credits because, otherwise, the models would have not been applied at all, and this was by far the least desirable outcome.

Appendix C: Table of Model Specifications

Table 1
Model Specifications for All Models

Model Name	Max Input Length[1] (Tokens)	Number of Parameters (Millions)	Encoder/Decoder Layers	Training Set Size (Articles)	Validation Set Size (Articles)	GPU Used for Training[2]
BART Base	1,024	140	6/6	8,000	2,000	Tesla K80
BART Large	1,024	400	12/12	8,000	2,000	Tesla K80
BART Large Pre-trained on CNN Daily Mail	1,024	400	12/12	8,000	2,000	Tesla K80
BigBird PEGASUS Pre-trained on ARXIV	4,096	576	16/16	2,000	500	A100
BigBird PEGASUS Pre-trained on Big Patent	4,096	576	16/16	2,000	500	A100
Distil BART 6-6 Pre-trained on CNN Daily Mail	1,024	230	6/6	8,000	2,000	Tesla K80
Distil BART 12-3 Pre-trained on CNN Daily Mail	1,024	255	12/3	8,000	2,000	Tesla K80
Distil BART 12-6 Pre-trained on CNN Daily Mail	1,024	306	12/6	8,000	2,000	Tesla K80
Distil BART 12-1 Pre-trained on XSum	1,024	222	12/1	8,000	2,000	Tesla K80
PEGASUS Large	1,024	568	16/16	2,000	500	A6000
PEGASUS Pre-trained on ARXIV	1,024	568	16/16	2,000	500	A6000
PEGASUS Pre-trained on CNN Daily Mail	1,024	568	16/16	2,000	500	A6000
T5 Base	512	220	12/12	8,000	2,000	Tesla K80

T5 Small	512	60	6/6	8,000	2,000	Tesla K80
T5 Small Pre-trained on WikiHow	512	60	6/6	8,000	2,000	Tesla K80

[1] Max input length controls the length of the padding and truncation.
[2] In order to use GPUs to train the model, CUDA (version 11.6) is used. Different GPUs are used because lack of availability and resources.
[3] All models are uploaded to Hugging Face (https://huggingface.co/Kevincp560)
[4] All models are trained for 5 epochs.
[5] All models use a learning rate of $2e^{-5}$.
[6] All models use a batch size of 2.
[7] All models are optimized using Adam with betas = (0.9, 0.999) and epsilon = $1e^{-8}$.

Appendix D: Table of Evaluation Results

Table 2
Evaluation Results of All Models After 5 Epochs

Model Name	ROUGE-1[1]	ROUGE-2[2]	ROUGE-L[3]	ROUGE-L Summary[4]	Gen Len[5]
BART Base	9.39	4.05	8.45	8.97	20.00
BART Large	10.95	5.09	9.56	10.43	19.05
BART Large Pre-trained on CNN Daily Mail	40.49	16.74	24.98	36.40	142.00
BigBird PEGASUS Pre-trained on ARXIV	**45.48**	**20.02**	**28.36**	**41.46**	**219.14**
BigBird PEGASUS Pre-trained on Big Patent	45.09	19.55	27.39	41.11	231.61
Distil BART 6-6 Pre-trained on CNN Daily Mail	39.28	15.88	24.23	35.27	141.86
Distil BART 12-3 Pre-trained on CNN Daily Mail	40.56	16.98	25.34	36.46	141.95
Distil BART 12-6 Pre-trained on CNN Daily Mail	40.10	16.50	24.83	36.08	141.88
Distil BART 12-1 Pre-trained on XSum	27.00	12.73	19.87	25.05	59.97
PEGASUS Large	39.11	15.41	24.37	35.12	226.59
PEGASUS Pre-trained on ARXIV	44.29	19.05	27.11	40.26	230.59
PEGASUS Pre-trained on CNN Daily Mail	37.26	15.82	24.20	34.03	125.89
T5 Base	9.38	3.70	8.49	9.00	19.00
T5 Small	8.83	3.26	8.00	8.45	19.00

T5 Small Pre-trained on WikiHow	8.96	3.27	8.15	8.57	19.00

[1] ROUGE-1 refers to the algorithm's and reference summaries' overlap of unigrams.
[2] ROUGE-2 refers to the algorithm's and reference summaries' overlap of bigrams.
[3] ROUGE-L computes the longest common subsequence (LCS) between a candidate and reference summary while ignoring new lines of text.
[4] ROUGE-L Sum computes the LCS between each pair of candidate and reference sentences while including new lines of text as sentence boundaries.
[5] Gen Length refers to the average length of the summary (in words) generated by the algorithm.
[6] The scores on the ROUGE metrics range from 0 to 100.