Abstract Text Summarization using PySpark and PEGASUS

Reg Gonzalez
Erik Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Richardson, Texas
rdg170330@utdallas.edu

***Abstract*—As advances in artificial intelligence, machine learning, and more specifically, natural language processing continues (NLP), one of the most important tasks is text summarization. Being able to extract key ideas from large documents is fundamental in these areas of computer science, as many of them rely on that kind of data. This project will attempt to perform abstract text summarization using PySpark and an NLP model called PEGASUS. The documents in this project that will be used to test this model include the book *Letters of Demonology and Witchcraft* by Walter Scott and an excerpt of the *Joker: Folie à Deux* Wikipedia page. The effectiveness of PEGASUS will be based on a subjective review of the summary it provides as well as an objective review of its ROUGE scores—scores that compare the similarity of two different pieces of text. Further research can be done by comparing PEGASUS to other NLP models and utilizing different combinations of parameters and parameter values.**

I. INTRODUCTION

A. PySpark and PEGASUS

PySpark, as defined by the Apache Spark documentation, is the "Python API for Apache Spark. It enables you to perform real-time, large-scale data processing in a distributed environment using Python" [1]. Some important features of PySpark include RDDs (Resilient Distributed Datasets), which are objects that can be parallelized and processed at the same time; DataFrames, which are other objects that are used as structures to hold data; and libraries like MLib and GraphX, which are used for machine learning and graph processing tasks respectively [2]. Overall, these features, and many others, make it helpful for many data scientists, data engineers, and developers to work with big data and other types of machine learning projects.

There are two different types of text summarizations in NLP: (1) extractive summarization and (2) abstractive summarization. The former takes sentences straight for the source and uses them to form a summary that details the most important points of the document. The latter is more akin to how humans summarize documents. That is, we don't just pick out the most important sentences from the text and use that for the summary. Instead, we read the document, comprehend it, and form a summary of brand-new sentences—in our own words—that best represent what the document is saying [3]. PEGASUS, which stands for "Pre-training with Extracted Gap-sentences for Abstractive Summarization" is an NLP model used for abstractive summarization. According to Peter Liu and Yao Zhao, writers for the official PEGASUS paper, PEGASUS uses

"transformer encoder-decoder models to improve fine-tuning performance on abstract summarization." These models are also used with pre-training like GPT-2 and BERT [4].

B. Documents Chosen for Dataset

The documents chosen for this project are the book *Letters on Demonology and Witchcraft* by Walter Scott and an excerpt of the *Joker: Folie à Deux* Wikipedia article. Since one of the goals of this project is to work with big data, summarizing an entire book seemed like an interesting challenge. There was no particular reason why I chose *Letters on Demonology and Witchcraft* other than I thought it sounded like an intriguing book title. Besides, regardless of the book, the PEGASUS model still would've worked the same way.

I also wanted to choose at least one more source to apply this model on, which is where the Wikipedia article comes into play. The reason why I chose the *Joker: Folie à Deux* article specifically is much like my reasoning for choosing the book: it's simply an article about a movie I'm interested in. Regardless, the model still would've done the same work even if I chose another article. However, the reason why I'm only testing an excerpt of the article is to preserve storage space. Loading in and summarizing an entire book takes a lot of processing time and space, and when I would try to run the model on another book in the same Databricks notebook, it'd often crash. Because of that, I've chosen to simplify my second document and instead just make it an excerpt.

II. PROBLEM DESCRIPTION

The goal of this project is to generate some abstract text summary based on two different documents: *Letters on Demonology and Witchcraft* and the *Joker: Folie à Deux* Wikipedia excerpt. To generate this summary, the NLP model PEGASUS will be used. This model was chosen because it's a recent state-of-the-art model that has been proven to give good and accurate results for abstract text summarization. This project will also be done via PySpark in a Databricks notebook. The use of PySpark and Databricks will not only help with the big data requirements of this project, but it'll make the program very easy and simple to run.

We will look at two different metrics to evaluate the effectiveness of the PEGASUS model. The first will be a subjective review of the summary. I'll review the summaries generated and see whether or not they accurately represent the contents of the two documents. The second metric will be ROUGE scores. These scores compare the similarity of two different pieces of text. For this, we'll use a reference summary for both documents and compare that to the summaries generated by PEGASUS.

III. TECHNIQUES AND ALGORITHMS USED

A. PySpark Implementation

This program was developed via PySpark in a Databricks notebook. Installations of various libraries were necessary (e.g., sentencepiece, rouge-score, requests, etc.) in order to build and evaluate the model.

Other PySpark-specific implementations included the use of RDDs and DataFrames to hold the documents. In the fourth code cell of the notebook, the *Letters on Demonology and Witchcraft* text file was collected from GitHub—where it was publicly hosted—converted into an RDD, and then into a DataFrame. RDDs and DataFrames, as previously mentioned in Section I ("INTRODUCTION"), are essential features of PySpark. They can hold unstructured data in a structured format, can be processed in parallel, and can have a variety of operations applied to them [5].

Beyond the use of RDDs and DataFrames,

which is prevalent throughout the rest of the code, the other important PySpark implementation lies in the fifth code cell of the notebook. Part of this code broadcasts the tokenizer and model (both from the PEGASUS model) to each worker in the cluster. This piece is imperative for two reasons. Firstly, it allows each worker in the cluster to process the text and create summaries. Secondly, broadcasting allows each worker node to receive a copy of the dataset (in this case, a book or excerpt from a Wikipedia article) by keeping a read-only variable cached for each worker. This makes it much more efficient because otherwise, you'd be passing the objects to each worker in the cluster, which can cause high overhead [6].

The last important PySpark implementation can also be seen in the fifth code cell, which is the registration of a user-defined function (UDF) in Spark. This code in particular registers a function called "text_summarization_udf," that returns the results of another function called "summarize_book_chunks," that, as the name suggests, returns a summary of the various chunks of text from the two input documents. This code was important because it was needed to work with DataFrame columns, which is the data structure we're using.

B. PEGASUS Implementation

The meat of this program is working with the PEGASUS model. There are some preliminary installations that need to be done, which are performed in the first four code cells, but the true work begins in the code cell #5. First, we pick the model that we want to use, and since we're using PEGASUS, that is the model we pick. Secondly, we call the PEGASUS tokenizer that corresponds with the model we picked. Since it is a tokenizer, it will convert the text data we gave it into integer tokens. Third, we define an instance of the model, which will ultimately be responsible for generating the summary based on the input text [3].

In the same code cell there is the method "summarize_book_chunks," which was mentioned earlier. This method does the bulk of the summarization using the tokenizer and model that we defined earlier. The most significant portions of the method are located in the for loop. Here, we summarize each chunk of text using numerous methods and passing in a variety of parameters. The first line converts the book chunks (or in the case of the *Joker: Folie à Deux* excerpt, text chunks) into inputs that are suitable for the PEGASUS model. The second line generates IDs of tokens for each chunks' summary. These chunks will then be decoded to get the actual, generated text summary. An overview of the parameters will be discussed in section IV ("RESULTS AND DISCUSSION"), since changes in those parameters are what I tested and evaluated. Lastly, once the tokens are decoded and the text summaries for each chunk are collected, they are appended to a list of text summaries. As you can see, the actual work done to generate the summaries is left to the PEGASUS model itself; that is not something we are responsible for [3].

In regards to how PEGASUS works, it essentially removes/masks important sentences from the input text and generates one output sentence based on the remaining sentences of the document [7]. During the pre-training phase, the sentences that were removed/masked need to be recovered by the NLP model so that the model can output those [concatenated] sentences as part of the summarization task. In general, this is a difficult task because you need to evaluate the sentences and determine which ones are "important." To do this, the creators of PEGASUS used ROUGE scores [5].

Going back to the code, once the summaries are collected, we can print them out to see what the PEGASUS model generated. However, just printing out the summary might not be sufficient. Because of that, we also chose to calculate and implement ROUGE scores.

ROUGE scores are an evaluation metric that compares two pieces of text by calculating overlapping n-grams. There are three different types of ROUGE scores: ROUGE-1, ROUGE-2, and ROUGE-L [5]. These scores range from [0,1], with 0 representing no overlapping n-grams and 1 representing a text full of overlapping n-grams [8]. Based on an objective overview of these scores, as well as a subjective review of the summaries generated by the PEGASUS model, we can evaluate whether or not the summaries were representative and accurate to the input text documents. A more in-depth discussion of these scores will be done in the following section.

IV. RESULTS AND DISCUSSION

A. Overview of Evaluation Metrics

Before we discuss the results of both the *Letters on Demonology and Witchcraft* and *Joker: Folie à Deux* texts, note that that Figure 1 is not a holistic representation of the results of this project. It is a subsection of the results that represents the best outcomes of the summaries generated for each respective text. To view the full results of this project, refer to the document called "CS 6350 Project – Results."

ROUGE scores, as previously indicated, were the objective evaluation metric that we went with. This is because PEGASUS model uses them to determine what sentences are "important" in the input text. ROUGE scores calculate overlapping n-grams in a text. According to the website Dremio, n-grams are simply "a contiguous sequence of words, symbols, or numbers extracted from a text document" [9]. We touched on three different kinds of ROUGE scores: ROUGE-1, ROUGE-2, and ROUGE-L. ROUGE-1 represents overlapping unigrams (one word), ROUGE-2 represents overlapping bigrams (two words), and ROUGE-L takes into account Longest Common Subsequence (LCS), which doesn't necessarily take into account the consecutive positions of text sequences within the original document. These scores have a range from 0 to 1, but a "moderate" score of ROUGE-1, ROUGE-2, and ROUGE-L is between 0.4-0.5, 0.2-0.4, and 0.3-0.4 respectively. Anything above those scores would be considered good for each respective type of ROUGE score [10].

While ROUGE scores can be a good objective representation of how accurate these generated summaries can be, what's equally as important is getting a subjective overview of what these summaries are actually saying. Calculating overlapping n-grams is great, but at the end of the day, PEGASUS is an *NLP* model, and as the name suggests, it is supposed to be characteristic of natural language. And who better to evaluate such language than a human? Simply looking at and reviewing the summaries from a human perspective provides a different perspective rather than plaining looking at numbers. For example, you can catch mistakes in the summaries in a way that ROUGE scores cannot since they only focus on calculating overlapping n-grams. This, combined with the ROUGE scores, are the primary evaluation metrics that were used in this project.

B. Parameters Used

When building the PEGASUS model, there are a few parameters that I chose to use: early_stopping, length_penalty, num_beams, min_length, and max_length. There are many other parameters that could've been chosen and evaluated, but keeping to these five made it simpler to evaluate.

early_stopping set to "True" means that the summarization process will stop if the end-of-sequence token is generated before the parameter max_length is reached. length_penalty, as the name states, controls the penalty based on the length of the text. Larger values for this parameter indicate a larger penalty, ultimately incentivizing shorter summaries. num_beams sets the number of

beams for beam search. Beam search is a type of algorithm that explores different possibilities for sequences during the summarization process. For example, num_beams was set to 10, then it'd keep track of the top 10 sequences that are most likely to occur at every step of the summarization process and ultimately pick the best one. Finally, min_length and max_length are self-explanatory. They represent the minimum and maximum length of the summary, respectively [11].

C. "Letters on Demonology and Witchcraft" Results

Firstly, I want to mention that in the experiments for both this and the *Joker: Folie à Deux* article, I only changed one parameter at a time. I had a "default" set of values I wanted to test for each parameter and I would only change one of them at a time (for example, just changing early_stopping while keeping all other parameters the same). I did this for two reasons: (1) it'd be impractical to test all the possible permutations of the parameters, and (2) I wanted to see the effects changing one parameter had on the summary and ROUGE scores.

The summaries that were generated by PEGASUS can be seen in the "CS 6350 Project – Results" document. For the sake of brevity, I'm not going to include them here in this report. Overall, most of the summaries were similar and mainly differed in regards to the length. Remember that PEGASUS outputs *one* output sequence/sentence for the summary. In terms of which parameters had the biggest effect on the summaries, that was early_stopping and num_beams. In the "default" values, early_stopping was set to True, but when I changed it to False, the length of the summary unsurprisingly increased. However, the actual content was nothing substantive. As you can see in Experiment #2 in the "Results" document, the same phrase keeps getting repeated over and over, and in fact, the summary doesn't come to a definitive end. It stops because it hits the max_length value of 150. The change to the num_beams parameter also caused significant changes. I got this source from Project Gutenberg, and the first sentence in the file references the website. As you can see in Experiments #4 and #5, the summaries make reference to Project Gutenberg. I explained earlier that num_beams controls the Beam search algorithm, which looks at different possible sequences for the summary generation. In this case, changing the num_beams parameter made it so that the other possible sequences, which evidently included references to Project Gutenberg, could be output.

Experiment #	**Text Used**	**Parameters**	**ROUGE Scores**
1	*Letters on Demonology and Witchcraft* by Walter Scott	early_stopping = True length_penalty = 2.0 num_beams = 5 min_length = 50 max_length = 150	ROUGE-1: Precision = 0.268 Recall = 0.25 F-Measure = 0.259 ROUGE-2: Precision = 0.025 Recall = 0.023 F-Measure = 0.024 ROUGE-L: Precision = 0.195 Recall = 0.182 F-Measure = 0.188

10	*Joker: Folie à Deux* Wikipedia article	early_stopping = True length_penalty = 0.5 num_beams = 5 max_length = 150	ROUGE-1: Precision = 0.636 Recall = 0.171 F-Measure = 0.269 ROUGE-2: Precision = 0.3 Recall = 0.075 F-Measure = 0.12 ROUGE-L: Precision = 0.455 Recall = 0.122 F-Measure = 0.192

Fig. 1. Table showing the best ROUGE scores for each respective input text

The summaries for *Letters on Demonology and Witchcraft* were all fairly accurate, except for the experiments that included references to Project Gutenberg. I'd say that experiments #1, #3, and #6 provided the best summaries that accurately represent what the book is about.

Now let's turn our attention to the ROUGE scores. Across all the experiments for this text, the ROUGE scores were not great. As a refresher, a "moderate" range for ROUGE-1 is 0.4-0.5, for ROUGE-2 it's 0.2-0.4, and for ROUGE-L it's 0.3-0.4. Even among the best of the experiments (Experiment #1), which can be seen in Figure 1, the scores do not reach those thresholds. Why is that? Well, let's remember that ROUGE scores compare two different pieces of text. The first one is obviously the summary generated from PEGAUS, but what is the second one? It's a reference summary. A reference summary is a summary obtained from an outside source that you know accurately represents the document you're summarizing. The one that I got was from the Walter Scott Digital Archive, run by the Edinburgh University Library [12]. This is a reliable source and had a synopsis of *Letters on Demonology and Witchcraft*. I used their synopsis of the book and compared it to the summaries created from PEGASUS. The reference summary is very different from the summaries formed by PEGASUS. It uses very different language overall and doesn't have the same unigrams, bigrams, or longest common subsequence as PEGASUS' summaries. It's because of that, that the scores are so low. However, as I stated earlier, that's why it's important to have a subjective review of the summaries as opposed to simply relying on the ROUGE scores. If you look at the reference summary and compare it to the summaries generated (again, both in the "Results" document), you'd see that they roughly say the same thing, just in different ways. They both mention that the book is a collection of letters from Walter Scott about topics of witchcraft, demonology, and the occult.

However, if you want to improve these scores, you can either choose a different reference summary or you can play around with the parameters more. Regardless of the low ROUGE scores, I'd say that PEGASUS did a decent job at accurately summarizing *Letters on Demonology and Witchcraft*.

D. "Joker: Folie à Deux" Results

The main difference between the *Letters on Demonology and Witchcraft* and *Joker: Folie à Deux* experiments is that the latter has one less parameter to work with: min_length. I chose to omit this parameter because the text I'm using is

much smaller than an entire book. Other than that, I've kept all the other parameters.

Similar to how Experiment #1—seen here in this report and also in the "Results" document—had my "default" values for each parameter, Experiment #8 had my "default" values for the *Joker* text. As seen in that document, the experiments after #8 include changes to one parameter at a time so, like with the first set of experiments, we can compare what effects changes in one parameter had to the "default" values.

First, let's take a look at the summaries themselves. Overall, the summaries were comparable across the board. All but one of them (Experiment #10) said something about the casting of the movie. Some of them had more details than others, for example, experiments #8 and #9 included the most information about the casting. That can be attributed to the fact that #8 had relatively high num_beams and relatively low length_penalty values. #9, on the other hand, had the early_stopping parameter set to False, which obviously plays a role in the length in the generated summary. The summaries, for the most part, accurately described the contents of the Wikipedia article. However, there was a detail many of them got incorrect. In experiments #8, #9, and #11, they say that actress Zazie Beetz is playing either Lee or Harley Quinn's music therapist; however, neither are true. In the movie, Harley will not have a music therapist and the character of "Lee" is played by Lady Gaga, not Zazie Beetz. All summaries also made references to either a "first trailer" or "first photo" being shown. *Joker: Folie à Deux* is a movie that is not currently in theatres, so they did release a first trailer and show first photos of the film relatively recently. What's fascinating about this is that the input excerpt from the article makes no mention of a first trailer or first photo. Perhaps this indicates that PEGASUS utilizes some real-time data to generate summaries.

Like with *Letters on Demonology and Witchcraft*, the ROUGE scores for *Joker: Folie à Deux* did not meet the "moderate" expectations of what the ROUGE scores should be. The only experiment that had some scores that could be described as "moderate" or "good" is Experiment #10, which is shown in Figure 1. ROUGE-1 should have a range of 0.4-0.5 to be "moderate," and while the Experiment #10's precision of ROUGE-1 is actually well above that range, its recall and F-Measure fall under. Similarly, ROUGE-2 should at least have a range of 0.2-0.4 to be "moderate," and while the precision of Experiment #10's ROUGE-2 score is in that range, the recall and F-Measure are not. The same goes for ROUGE-L. Experiment #10's ROUGE-L precision is above the "moderate" range of 0.3-0.4, but the recall and F-Measure values leave more to be desired. Purely looking at ROUGE scores, this experiment without a doubt had the best results; however, what's interesting is that the summary is arguably the *worst* out of all of them. The summary generated simply states "The first trailer of Joker: Folie a Deux has been released." Unlike the other experiments' summaries, there's no mention of the cast or any other pertinent details of the film. So why is the ROUGE score so why? Well, it's because of how short the sentence is. When we compare Experiment #10's summary to the reference summary (seen in the "Results" document) for *Joker*, the phrase "Joker: Folie a Deux" is seen in both pieces of text. Because #10's summary is so short, when we calculate ROUGE scores, the proportion of overlapping n-grams to total words in much higher. This is why #10's ROUGE scores are high even if the summary itself is bad. Again, this is why having a subjective overview of the summaries is necessary for this kind of task.

V. CONCLUSIONS AND FUTURE WORK

As artificial intelligence and machine learning continue to advance, one key area of

development that has quickly evolved is natural language processing. Large language models (LLMs) continue to become more sophisticated and complex, generative AI content is more popular than ever before, named entity recognition is becoming more commonly used, etc.

However, one advancement in particular was the focus of this project: abstract text summarization. Throughout this project, we used PySpark and the PEGASUS model in order to summarize two different pieces of text: the book *Letters on Demonology and Witchcraft* by Walter Scott and an excerpt of a Wikipedia article of the upcoming movie *Joker: Folie à Deux*. PySpark is the premiere Python API that allows you to use Apache Spark—a distributed processing engine that works effectively for big data. PEGASUS, on the other hand, is an NLP model that takes in an input document and outputs a one-sentence sequence that represents a summary of what that document was about.

To evaluate whether or not PEGASUS was effective in document summarization, I decided to approach this in two different ways: (1) a subjective review of the generated summaries, and (2) an objective review of the ROUGE scores. Overall, the results were decent. While all but one of the experiments failed to meet the "moderate" ROUGE score requirements, the summaries themselves were not bad and were mostly representative of what the two pieces of text were about. There were some inconsistencies and repeated language in the summaries (depending on what the values were passed for each parameter), but the summaries were sufficient. In the end, it was found that Experiment #1 had the best parameter values for *Letters on Demonology and Witchcraft* and Experiment #10 had the best parameter values for *Joker: Folie à Deux*.

In regards to any future work, you could obviously continue the research by experimenting with the parameters more. For the sake of brevity, I chose to test one parameter at a time, but you could very well test out even more parameters or combinations of the same parameters with different values. Beyond that, I think that comparing PEGASUS to other NLP models and techniques could be interesting. For example, you could use BART or other tokenizers and transformers to perform abstract text summarization.

The field of artificial intelligence, machine learning, and NLP is vast and everchanging. There are constantly new things to explore and develop, and with technologies like PEGASUS and PySpark/Apache Spark we can create and discover new techniques to achieve once thought to be impossible goals.

REFERENCES

[1] "PySpark Overview — PySpark 3.4.1 documentation," *spark.apache.org*. https://spark.apache.org/docs/latest/api/python/index.html#:~:text=PySpark%20is%20the%20Python%20API

[2] Nagar, Anurag. "Spark MLlib." Class lecture, Big Data Management and Analytics, The University of Texas at Dallas, Richardson, Texas, August 1, 2024.

[3] "Text Summarization in NLP," *GeeksforGeeks*, Feb. 05, 2024. https://www.geeksforgeeks.org/text-summarization-in-nlp/

[4] P. Liu and Y. Zhao, "PEGASUS: A State-of-the-Art Model for Abstractive Text Summarization," *Google Research*, Sep. 6AD. https://research.google/blog/pegasus-a-state-of-the-art-model-for-abstractive-text-summarization/

[5] Nagar, Anurag. "PySpark DataFrames." Class lecture, Big Data Management and Analytics, The University of Texas at Dallas, Richardson, Texas, August 1, 2024.

[6] "Broadcast," *spark.apache.org*. https://spark.apache.org/docs/1.5.1/api/java/

org/apache/spark/broadcast/Broadcast.html (accessed Aug. 01, 2024).
[7] “Pegasus,” *huggingface.co*. https://huggingface.co/docs/transformers/en/model_doc/pegasus (accessed Aug. 01, 2024).
[8] “Evaluate translation or summarization with ROUGE similarity score - MATLAB rougeEvaluationScore,” *www.mathworks.com*. https://www.mathworks.com/help/textanalytics/ref/rougeevaluationscore.html
[9] “What is N-grams in NLP? | Dremio,” *www.dremio.com*. https://www.dremio.com/wiki/n-grams-in-nlp/#:~:text=An%20N%2Dgram%20is%20a (accessed Aug. 01, 2024).
[10] S. Walker II, “What is the ROUGE Score (Recall-Oriented Understudy for Gisting Evaluation)?,” *KLU*. https://klu.ai/glossary/rouge-score (accessed Aug. 01, 2024).
[11] “Generation,” *huggingface.co*. https://huggingface.co/docs/transformers/en/main_classes/text_generation
[12] “Letters on Demonology and Witchcraft,” *Walter Scott*. http://www.walterscott.lib.ed.ac.uk/works/prose/witchcraft.html#:~:text=Back%20to%20top-,Synopsis,period%20to%20his%20own%20day. (accessed Aug. 01, 2024).