Article Summarization for COVID-19 Medical Documents with Natural Language Processing

Jack Grossi

Dept. of computer and information sciences

Fordham University

New York, U.S.A.

Jgrossi2@fordham.edu

**Abstract—** Can we create a summary of a complex document with the major points of the original document? For this project we will create a text summarization model that will return a summary of a complex medical document for the use by medical professionals. There are two main methods of text summarization: abstractive and extractive. Extractive reuses pieces from the original text to create a summary while abstractive writes an original summary and contains some understanding of the original document. For this task we will be creating an abstractive summary using Seq2Seq to generate abstracts for these scholarly articles.

Keywords—Natural language processing (NLP), neural network, long short-term memory (LSTM), word embedding, summary generation

# I. INTRODUCTION

Text summarization is the process of generating short, accurate, and most importantly readable summaries of longer and more complex texts. There are two main methods of text summarization: abstractive and extractive. Extractive reuses pieces from the original text to create a summary while abstractive writes an original summary and contains some understanding of the original document. In conjunction with the enormous number of medical papers that have been published in the last two years related to COVID-19, text summarization can be an incredibly powerful tool that enables medical professionals to quickly and accurately determine which scholarly article is relevant to them.

|  |
| --- |
| XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE |

In this research project, we adopted natural language processing (NLP) techniques to aid medical professionals with scholarly and research article summarization. We will be building a Seq2Seq model using Keras and train it using our own dataset. Based on the accuracy of our generated summaries, we can determine how effective our model is.

This paper is organized as follows. Section II our dataset, the preprocessing steps that were taken, and how the challenges we faced during this stage. Section III describes the model creation, and Section IV discusses the experimentations results. Section V covers the conclusion of this research and future work.

# II. METHODS

## A. Dataset

Our dataset will be the COVID-19 Open Research Dataset Challenge (CORD-19). This dataset was created in response to the COVID-19 pandemic by the White House and a coalition of leading research groups. It contains over 1,000,000 scholarly articles, with over 400,000 of those articles containing full text. The dataset is updated and maintained on a regular basis, adding, and removing full text documents. A large amount of effort will primarily be directed at the data preprocessing steps for this dataset as it is incredibly large and requires adjustments before any analysis or modeling is done. As the documents included in the dataset range from full scholarly papers to short abstracts, they must be sorted, and relevant documents selected.

The articles from this dataset are stored in JSON format in two categories: PDF and PMC. As the PDF JSON files contain full text and abstract, we will only be using these files for the purposes of this project. PMC files only contain the body of the articles and will not assist us in training a language model. From these PDF JSON files, our training set consists of 401,270 full articles.

## B. Data Handling & Preprocessing

As our dataset is 87 GB large, moving it between development environments is near impossible. Therefore, all development will be done in Kaggle’s notebook development environment. Additionally, after initial data loading took near an hour to open all JSON files, a mini batch was needed. This mini batch consisted of 10% of our dataset, or 40,000 articles. From this batch we were able to create a dataframe consisting of Paper\_id, Abstract, Body\_text, Methods, and Results. From this, we were able to generate a word count for each available abstract and body\_text which were added for each entry.

Our initial exploration revealed that some body\_text entries were non-unique, meaning that some articles had repeated text under the same ID. Table 1 outlines additional statistical findings.

|  |  |  |
| --- | --- | --- |
|  | Abstract Word Count | Body Word Count |
| Mean | 160.55 | 3983.7 |
| STD | 195.22 | 5042.96 |
| Min | 0 | 1 |
| 50% | 151 | 3285 |
| Max | 5349 | 241927 |

TABLE I. Data Exploration

The word counts were then plotted in Tableau to determine if there were and meaningful trends or visualizations that might be overlooked. This plot is displayed in Fig 1.

Chart, scatter chart

Description automatically generated

Figure 1. Visualization of Word Counts

As one can see, there is no real relation between abstract length and body length.

Our final step of data preprocessing before splitting into train and test sets was text cleaning. In this step, we removed stopwords, removed punctuation and extraneous characters, lowercase all words, removing contractions and restoring them to their original words (ex. “You’re” becomes “you are”), stemming suffixes from words, and lemmatizing all words in our corpus.

With this complete we are finally able to split our dataset into training and testing. However, due to memory constraints, training was not possible on our still massive dataset. Therefore, additional data sampling had to be done in order to reduce the dataset to a size able to run in Kaggle’s environment. To do this, we first dropped any extraneous columns that were not our paper\_id and cleaned texts. Next our dataframe was split into training and testing sets on an 80-20 split. From this, we randomly sample our training set, randomly selecting 50% of the entries to use. After this, our training set contains 11,280 entries and our test set contains 5,641 entries.

# III. MODEL

Sequence-to-Sequence models are neural networks that take a sequence from a specific domain as the input and output a new sequence in another domain. [3] In this section we will outline the creation of the following characteristics of Seq2Seq models:

* Creating Sequences
* Word Embedding
* Encoder-Decoder Model
* Training and Inference Generation

## Feature Engineering

Feature engineering will encompass both the creation of the sequences as well as word embeddings. We begin by finding the most frequent words for both our X and Y in our training set; X being our body and Y being our abstracts. The topmost frequent words are outlined in figures 2 and 3 respectively.

Chart, histogram

Description automatically generated

Figure 2. Top Most Frequent Words in Body

Chart

Description automatically generated

Figure 3. Top Most Frequent Words in Abstracts

From this we can identify words that have a frequency below a certain threshold that we will consider statistically insignificant. We have determined this threshold to be 5, therefore all words occurring less than 5 time are dropped due to statistical insignificance. We can also see that our preprocessing was not perfect as words like “was” are truncated to “wa”. This should not have an effect on our final summary so long as we are able to translate it back ourselves.

Chart

Description automatically generatedOur next step is to create our feature matrices by transforming our preprocessed text into a list of sequences using Keras. We being by tokenizing our corpus using Keras’ tokenizer and then creating our sequences from those tokenized entries using texts\_to\_sequences. Finally, we pad our sequences using pad\_sequences. For our abstracts we follow the same steps after adding <START> and <END> tags to the beginning and end of each abstract. Figure 4 outlines how our word embedding process functions.

Figure 4. Word Embedding Example [1]

## Model Building

Diagram

Description automatically generatedOur model will follow an Encoder-Decoder structure where our encoder is fed an input. The decoder is then shown the y minus the end and the model attempts to predict how the y ends. In our case, the input is the cleaned text body and y is our cleaned abstracts. This type of training is “teacher forcing” which uses the targets rather than the output generated by the network so that it can learn to predict the word after the start token, then the next word and so on. [1] The structure of our model is depicted in Figure 5.

Figure 5. Encoder-Decoder Structure

Our model has a total of 31,185,097 trainable parameters.

## C. Inference Generation

In order to examine the outputs of our Seq2Seq model, we must create an Inference Model to generate predictions from our Seq2Seq model. This model also follows an Encoder-Decoder format where the encoder takes an input sequence of X\_test and returns the output of the last LSTM layer. The decoder then takes the start token and the output of the Encoder and returns the new states as well as a probability distribution over the vocabulary. This model is iterated over until the END token is predicted or the maximum length is reached.

# IV. RESULTS

In this section we will discuss how successful our model was at generating abstracts for complex scholarly articles, the different parameters used, and present examples for each stage.

The two main metrics we looked at when training the model were Validation Loss and Time. We knew we would never achieve the sort of accurate predictions found in pre-trained models, so we instead chose to focus on how effective we could make our model and the time tradeoff necessary to do so. All runs were done using RMSProp as the optimizer and Sparse Categorical Crossentropy as the loss function. Batch Size was kept at 128 and a validation split of 0.25 was used.

Our first run of the model was set at 10 epochs which achieved a validation loss of 6.151 after only 12.5 minutes.

Chart, line chart

Description automatically generatedTraining Validation

As expected, the generated summaries were very poor for this run. Outputs had a tendency to being repeating a select word after approximately 10 words. Examples are included below:

Predicted summary:  background aim study aimed investigate clinical care outcome patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient patient

Predicted summary:  background aim study aimed investigate impact health care care care hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital hospital

While adequate results were not necessarily expected during this run, the repeating word was a result that was unexpected. To attempt to rectify this issue, our first adjustment made was increasing the number of epochs for training. Our next run then was identical to the first run except with 50 epochs instead of 10. This run achieved a validation loss of 5.5102, a significant improvement over our first run. However, it took 60.5 minutes to run this model, a 384% increase.

Diagram

Description automatically generated with low confidenceTraining Validation

When summary predictions were analyzed, results were significantly better than the first run, however, they were still not adequate enough to be used as proper summaries. Examples are included below:

Predicted summary:  background objective study wa ass impact patient associated among patient undergoing method retrospective study wa conducted retrospective study included study included study wa conducted using sampling tool method descriptive analysis wa used ass association variable correlation variable variable variable variable relationship

Predicted summary:  study aim explore whether knowledge attitude toward social medium among student n year n 18 year old participant completed questionnaire wa administered participant randomly assigned online questionnaire wa sent using questionnaire wa used collect questionnaire wa used collect data result total participant mean age participant satisfied satisfied video video student satisfaction online survey wa used collect data participant n 90 year old participant satisfied satisfied interview online survey wa used collect data

As one can see, results are better in the sense that summaries are now more complete sentences rather than just a repeating word. However, sentences are incoherent and still tend to be repeat words or phrases. There are standouts that show promise though, like the first few lines of Predicted Summary 2. While results improved significantly from run 1, it is still far from the desired outcome.

For our final run, rather than increasing the number of epochs, we instead increased the length of the sequences being fed into the model; our hypothesis being that longer sequences would allow the model to make more informed predictions. This run achieved a validation loss of 5.5102, identical to our previous run. However, training took 144.5 minutes to complete.

Chart

Description automatically generated

Examples are as follows:

Predicted summary:  background covid19 pandemic ha led unprecedented challenge healthcare system worldwide pandemic ha affected country world health organization organization wa declared pandemic pandemic ha affected country world health organization organization africa country africa country africa country ha impact pandemic ha become increasingly affected pandemic due pandemic impact covid19 pandemic ha impact pandemic ha affected country world health organization pandemic ha brought unprecedented challenge covid19 pandemic

Predicted summary:  background study aimed investigate effect risk factor among patient covid19 method retrospective study wa conducted using validated data collection data collected using online questionnaire wa conducted using online questionnaire wa used collect data data collected using spss version wa used analyze data result total age patient included study included study wa conducted using spss version wa used analyze data result total patient included study included study wa conducted using data wa used determine frequency

Results appear to have little improvement over the previous run with an increase to sequence length having little impact at 50 epochs.

# V. CONCLUSION AND FUTURE WORK

In this research project, we showed interesting findings creating a Natural Language Processing model for complex scholarly article summarization. If work were to be continued on this project, three main improvements would be made. The first is training on a larger number of epochs. As we saw, increasing the number of epochs made a dramatic difference in the prediction capabilities of the model. The second is implementing a language detection feature that is able to sort out non-english articles. Despite the preprocessing steps taken, some non-english articles were able to make it into the final training set which can throw off model performance. Finally, the number of trainable parameters could be increased which may improve the model’s performance. However, for all of these improvements, greater computation power would be needed as Kaggle becomes a limiting resource for extremely large datasets and models.

# REFERENCES

1. Pietro, Mauro Di. “Text Summarization with NLP: TextRank vs Seq2Seq vs Bart.” Medium, Towards Data Science, 16 Mar. 2022, <https://towardsdatascience.com/text-summarization-with-nlp-textrank-vs-seq2seq-vs-bart-474943efeb09>.
2. Alla, Samhita. “Implementing Seq2Seq Models for Text Summarization with Keras.” *Paperspace Blog*, Paperspace Blog, 9 Apr. 2021, <https://blog.paperspace.com/implement-seq2seq-for-text-summarization-keras/>.
3. <https://en.wikipedia.org/wiki/Seq2seq>
4. **Chollet, F.** **& others, 2015.** **Keras**. Available at: <https://github.com/fchollet/keras>.