TEXT SUMMARIZATION USING RECURRENT NEURAL NETWORKS

Arpita Ajit Welling, Harshitha Biligere Harish, Sanika Anil Paranjpe
aawellin@iu.edu hbilige@iu.edu sparanjp@iu.edu

Luddy School of Informatics, Computing, and Engineering,
Indiana University Bloomington, April 2022

ABSTRACT

Text summarization is a ubiquitous application. It can be used for news summarization, product review, youtube video title generation and more. In this project, we propose to build a model that summarizes a given document or a given text paragraph. Literature states a lot of methods to do this. We plan on using a neural network approach. Our plan is to use a Recurrent Neural Network with gated recurrent unit (GRU) and compare the results with the Long Short Term Memory(LSTM) model for the same datasets. We propose to do text summarization on CNN/Daily News dataset and extend it on Amazon Fine Food Reviews.

Index Terms— Natural language processing, Recurrent Neural Networks, CNN/Daily News, Machine Learning

1. INTRODUCTION

In today's exponentially growing world of technology, the consumption and application of data collected has led to the necessity of summarization of information for better analysis. Manually summarizing the information is a tedious and time consuming task. This is where the text summarization model comes in. Summarization, in general, refers to pruning and presenting information in a shorter version, whilst preserving the core meaning and key information of the original data. The main focus of text summarization, specifically, is to generate a subset of relevant and important information from the entire set of raw data.

Since human interaction is non-existent in automated text summarisation, it lacks human knowledge and language capability, making it a challenging and arduous job. As a result, automatization of text generation has gained traction, becoming a prominent field of study in natural language processing. It has also motivated research on the optimal way to achieve summarization and its various applications. The applications range from search engine optimizations, news classification, legal document summarization, headline generation, scientific report summarization and so on.

Automatic text summarization can be achieved with two different approaches - extractive summarization and abstractive summarization. Extractive summarization fetches sentences directly from the document based on a scoring function and forms an articulate summary. Abstractive summarization, on the other hand, creates a bottom-up summary by interpreting the data to generate a new summarized text. Parts of the information present in the new text may not be present in the original text, requiring rephrasing sentences and grammatical syntax analysis.

In our project, we are taking an abstractive text summarisation approach. A sequence-to-sequence Recurrent Neural Network architecture is used to achieve this.

Furthermore, the model will be experimented on and address the following research questions:

- Can this model work on a different dataset like, amazon fine food reviews dataset?
- How well does our GRU model perform compared to other architectures like LSTM and also how well the model performs compared to already existing libraries?
- How close are our predictions to the actual summary given in the training dataset?

2. RELATED WORK

We read and analyzed the work done by NLP experts related to text summarization in order to complete this project.In this project we are comparing text summarization using LSTM and GRU using 2 different datasets to develop a comparative study. We studied both extractive and abstractive text summarizers in order to understand the working of a text summarizer. There is a lot of work in both domains of text summarization.

There can be different ways of developing a text summarizer. Traditional neural networks can be used or Recurrent neural networks can be used.

Authors in paper [4] adopted the way of developing an Extratice type of text summarizer using traditional neural networks with one input, one hidden and one output layer.They have experimented on varying the page length and found Rouge scores for those page lengths.

Authors in paper [2] have developed an Extractive text summarizer by creating their own novel model called SummaRuNNer. They achieved a Rouge1 F1 score of 39.6 %.

Abstractive text summarization model from [1] does a comparative study on multiple datasets using GRU architecture used along with attention mechanism. Their model uses a hidden state size of 200 and uses a 100 dimensional embedding layer. We wanted to experiment with these values and thus ended up using 256 as hidden state size and embedding dimensions. They obtained a Rouge1 F1 score of 35% on the "feats-lvt2k-2sent-ptr" model.

Model in paper [3] does experiments on the DUC 2004 and Gigaword dataset and trains it on a variety of different models. They have compared the Rouge score on 8 different models and ran it on two datasets. ABS+ model from their implementations works best on both the datasets.

3. METHODS

For this project, our input is a document or a text paragraph that is essentially a sequence of words. The output of our system will also be a sequence of words but it will be smaller than the input.

This problem is posed as a many-to-many Seq2seq problem. To implement this encoder-decoder architecture, we used a Recurrent Neural Network with Gated Recurrent Unit component and another architecture with Long-Short Term Memory component.

3.1. Encoder-Decoder Architecture:

An encoder-decoder architecture is shown below:

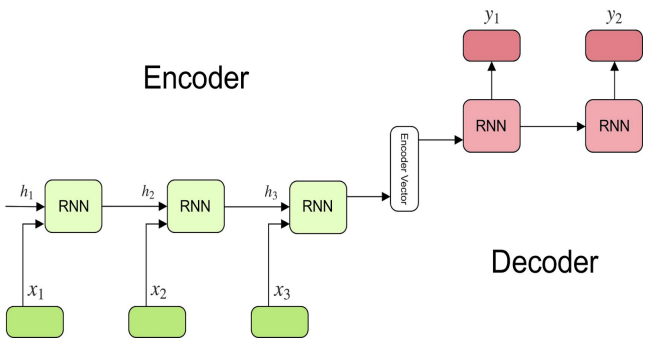

Fig. 1: Encoder-Decoder architecture (https://towardsdatascience.com/understanding-encoder-decoder-sequence-to-sequence-model-679e04af4346)

Here, x1,x2, and x3 represent the words in the input and y1,y2 represent the output words.

At each time-step, a new word from the sentence is given to the RNN unit along with the hidden state of the previous unit. Decoder gets output of the last encoder as input.

RNNs generally face the issue of vanishing gradients and exploding gradients. To overcome these issues, we use different models like LSTM and GRU. LSTM and GRU help to remember the long-term dependencies which RNN cannot do because of vanishing gradient problems.

3.2 Long Short Term Memory:

A LSTM unit is shown below:

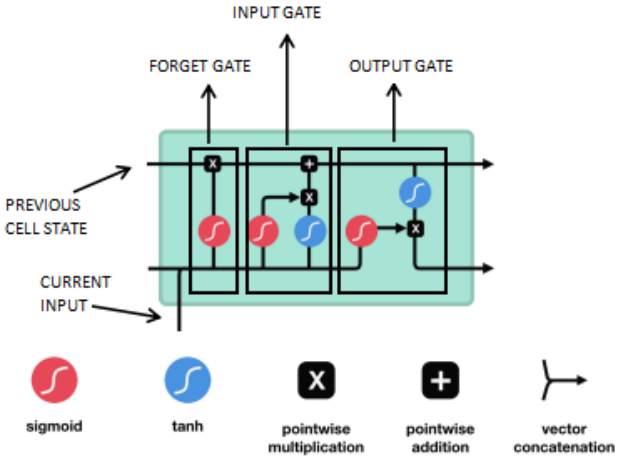

Fig.2: LSTM model architecture (https://medium.com/analytics-vidhya/lstm-whats-the-fuss-about-1ae9d4c3e33e)

The 2 main parts of the LSTM are the cell state and the forget state. The cell state decides which state to remember and which state to forget. The forget gate or the forget layer helps to do that. There is also an input gate layer (sigmoid) that helps to find new values for the cell state.

3.3 Gated Recurrent Unit (GRU):

A GRU unit is shown below:

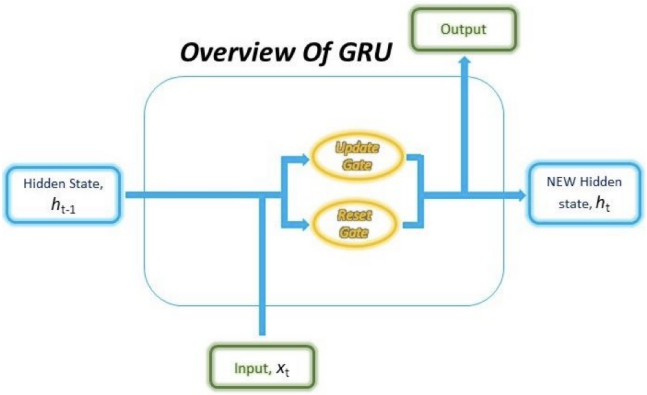

Fig.3: Overview of GRU (http://www.sefidian.com/2020/01/30/gated-recurrent-unit-gru-with-pytorch/)

The 2 main parts of GRU unit are update and reset gates. Update gate is used to decide which information to keep and

which information to throw while the reset gate is used to decide how much past information should be used.
The context vector is generated using a GRU. Each word is given as an input to a GRU and this outputs a hidden state which is passed on to the next GRU along with the next timestamp word input.

3.4 Data Preprocessing:

In articles, there are certain things that are written in brackets. These generally indicate acronyms or explanations of some things. While these things are important for the entire document, they are not that important for the summary or highlight of the article. Therefore, we have removed brackets and words or phrases from the brackets.
The most important thing in text data is separating the words into tokens. We have removed all punctuations from the data and separated the words as tokens.
Further, we have removed stopwords from the documents like articles (a,an,the), prepositions (between, in,before,after,etc.), pronouns (she,her,him,his,etc.) and possessive nouns. For simplicity, we are using lowercase for all words.
In the English language, many times we use contractions for certain words. For example, should have is written as should've and have not is written as haven't. We have expanded these contractions for our dataset. We have also removed words with length less than or equal to 2.
For textual data, stemming and lemmatization is done to transform the word to the root form. We are also doing stemming and lemmatization on our data.
For any natural language processing problem, we need to convert textual data to numbers. These numbers are then given to our model. This process is called tokenization. For example, if we have a sentence "Hello, What is your name?" tokenization will assign a number to each word. The sentence will be converted to [1 78 24 18 45]. These numbers are stored in a word index dictionary like {"hello": 1, "what": 78, "is": 24, "your": 18, "name";45}. This is also called sequencing data. We also use padding for these sequences to make all documents of the same length.

3.5 Optimizers:

We are using RMSProp Optimizer for our model. We chose this optimizer because it accelerates the process of optimization. It helps decrease function evaluations and reach the optima. It uses the decaying average or moving average of the partial derivatives to calculate the learning rate.

3.6 ROUGE score:

ROUGE score stands for Recall-Oriented Understudy for Gisting Evaluation. It is a set of evaluation metrics. The main evaluation metrics are ROUGE-1, ROUGE-2, ROUGE-3, precision, recall, and F-1 score. The ROUGE-N scores are calculated according to the N-grams model. Recall calculates the number of overlapping n-grams and divides it by the number of n-grams in the reference while precision calculates the number of overlapping n-grams and divides it by the number of n-grams in the model. F-1 score can be calculated by the following formula:
2*[(precision*recall)/(precision+recall)].

4. RESULTS

4.1 Datasets used:

The model has been run on and compared two datasets: CNN/daily news dataset with 280,000 and Amazon fine food reviews with 75,000 unique reviews. The CNN dataset has 2 main columns - articles and its highlights, and the Amazon fine food reviews has 'text' and 'summary' columns. There is no missing data in this dataset. Considering it is textual data, there are no outliers in it CNN dataset has an additional field 'id' which is irrelevant and hence dropped.

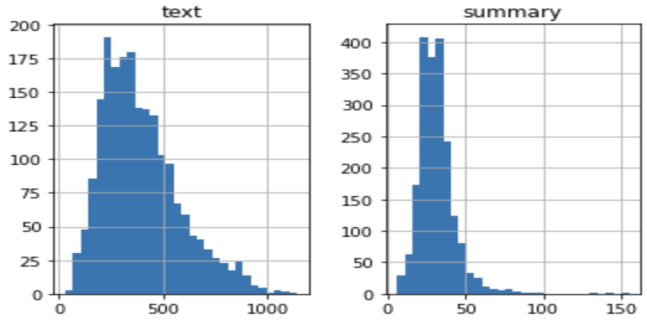

Fig.4: Bar graph representing lengths of article and summary for CNN dataset

4.2 Exploratory Data Analysis:

Each document in the CNN dataset will have different length and its highlight will have a smaller length. We have visualized the length for the first 10 articles and their highlights.
Every document has some parts of speech assigned to each word. In the image given below, we are visualizing the number of words in each POS for the first 10 documents.

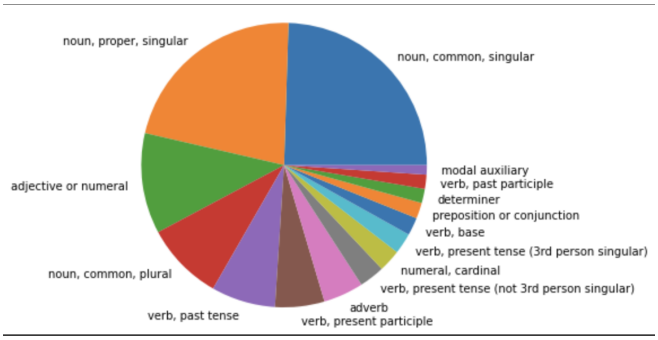

Fig.6: Pie chart representing distribution of 15 most frequent parts of speech

For visualizing documents, we have performed word cloud on the first document. The result of that can be seen below.

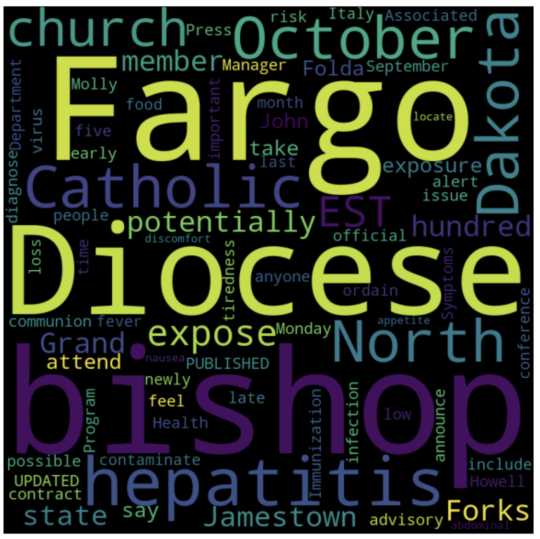

Fig.7: Sample WordCloud of 1st article

4.3 Experiments conducted:

Considering the dataset sizes are too big for computation on limited resources that are available to us, the experimentations are done only on 10k and 20k data for Amazon dataset, and 2k and 5k for CNN dataset. Training and validation accuracy and loss is calculated for each epoch (total no. of epochs = 50).

In general, it is observed that the training and validation accuracies of GRU models is better than LSTM. GRU gives accuracies of around 75% and loss of 1.9, whereas, LSTM gives accuracies around 68% and loss of 2.4.

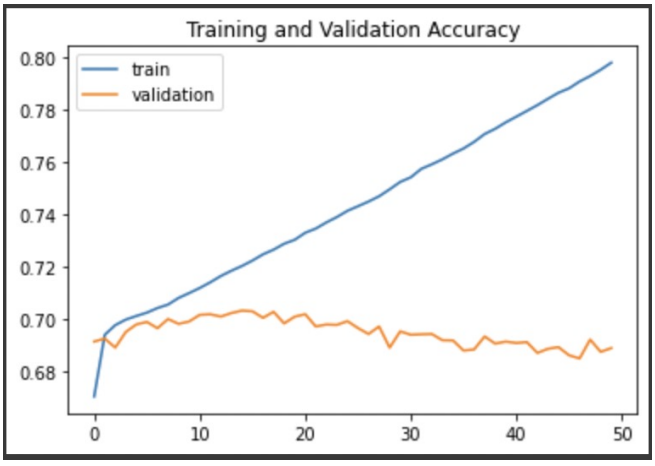

Fig 8: Training and validation accuracy for LSTM model on Amazon dataset for 20,000 data size.

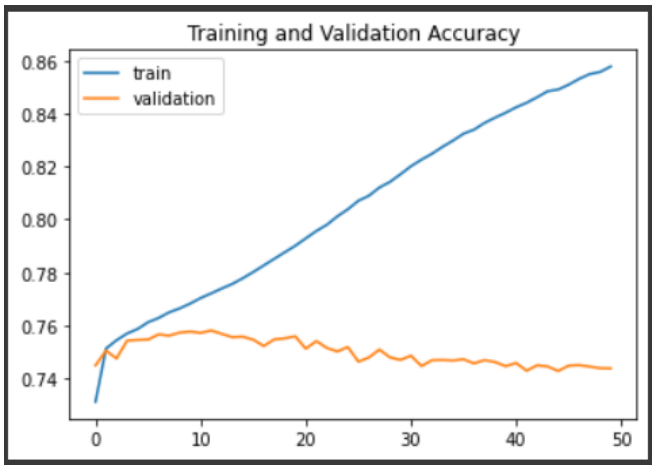

Fig. 9: Training and validation accuracy for GRU model on Amazon dataset for 20,000 data size.

From above graphs it is seen that we achieve higher training and validation accuracy on GRU model than the LSTM model.

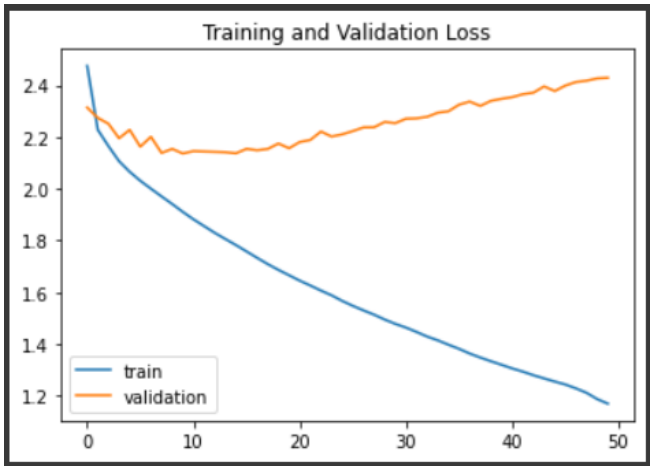

Fig. 10: Training and validation loss for LSTM model on Amazon dataset for 20,000 data size.

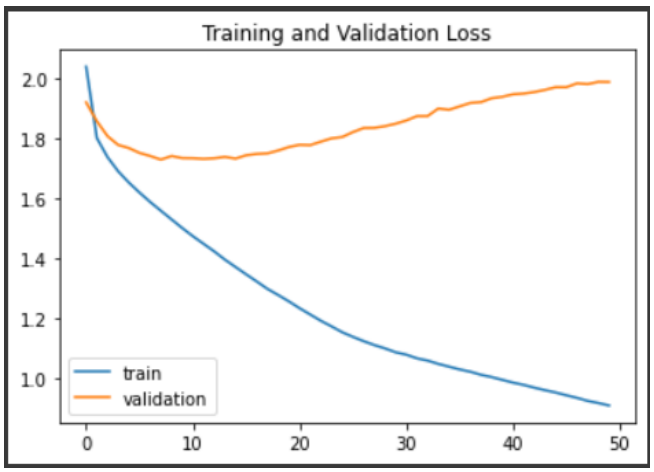

Fig. 11: Training and validation loss for GRU model on Amazon dataset for 20,000 data size.

ROUGE score is used for evaluating the text summariser model. Precision, recall and F1 scores are generated for each summary. We have taken the average of all these and presented the final scores.
Following are the average of ROUGE scores(p-precision, r-recall, f1 scores) for LSTM and GRU on Amazon and CNN dataset:

Model with data size ↓	Precision	Recall	F1 score
LSTM-2k	0.10	0.04	0.06
LSTM-5k	0.07	0.03	0.04
GRU-2k	0.08	0.04	0.06
GRU-5k	0.07	0.06	0.06

Table 1: ROUGE scores comparison for CNN dataset

Model with data size ↓	Precision	Recall	F1score
LSTM-10k	0.09	0.08	0.08
LSTM-20k	0.12	0.09	0.10
GRU-10k	0.10	0.08	0.08
GRU-20k	0.12	0.10	0.11

Table 2: ROUGE scores comparison for Amazon dataset

It can be inferred from the experiment tables that the GRU models behave better than the LSTM for both Amazon and CNN dataset..

Good Example of predicted highlights on Amazon data:

1)

Article: come across compel reason online add daily nutritional supplement buy product tell open jar skedish down tablespoon stuff first add oatmeal actually enhance flavor next move allow liquefy down tablespoon right jar delight wonderful taste
Actual Highlights: wonderful flavor smell and consistency
Predicted Highlighs: the perfect food

2)

Article: order twice year the come excellent condition within day order
Actual Highlights: great product great service seller
Predicted HIghlighs: excellent product

Failed Example:
Article: annie deluxe macaroni cheese quick prepare tasty speedy meal sometimes add cubed ham peas package another option bulk pricing amazon plus
Actual Highlights: annie deluxe macaroni cheese
Predicted Highlighs: never enough without sugar

5. CHALLENGES

The challenges faced while implementing the project are as follows:

1. As the dataset corpus is huge, the vocabulary created is also large, and a lot of time is taken, so we resorted to using a smaller data sample of 20k for Amazon fine food reviews and 5k for CNN dailymail. There was also a limitation on computational resources.
2. Implementation of the attention model led to some issues in the inference part, but we were successfully able to train the model with the attention layer.
3. TensorFlow version had to be downgraded to 1.15.2 from 2.8.0 for implementation of the attention layer.
4. We would have been able to produce better results with our model if we had not sampled the dataset for CNN data.

6. CONCLUSION

We have successfully implemented a text summariser on CNN dailymail and Amazon fine food review datasets. RNN models with both GRU and LSTM were used to create the model. We also proved that the GRU model behaves better than the widely used LSTM model.
Attention model could have been added to further improve the performance of the model in both LSTM and GRU.
Our implementation of this problem gave similar accuracy and scores to other implementations of this problem.
The research questions posed earlier by us have been answered through the implementation and experimentations in this project. This included using 2 different datasets with different sample sizes, implementation of LSTM model and proving that GRU is slightly better than traditional LSTM model, and that our model gives outputs similar to the expected summaries or human generated summaries. Although not perfect, the summaries generated are close to the human generated ones and are easy to understand.
In our future work, we would like to extend this problem and use attention models for better results. We would also like to apply the text summarizer to some useful daily life applications like legal document summarisation, search engine optimisation, video transcript summarisation and many more.
Overall, our final implementation lines up with our initial proposal. And all three of us really enjoyed working together on this project. We got insights into how the text is summarized and how it could be applied to different tasks. We also learned how recurrent neural networks are implemented and the details on encoder decoder

architecture. It was exciting to gain knowledge on LSTMs and GRUs and the way they both act as memory units but have different ways of storing this information. It was a great learning experience for us and will surely help us understand natural language processing tasks better.

7. REFERENCES

[1] Nallapati, R., Xiang, B., & Zhou, B. (2016). Sequence-to-Sequence RNNs for Text Summarization. CoRR, abs/1602.06023. Opgehaal van http://arxiv.org/abs/1602.06023

[2] Nallapati, R., Zhai, F., & Zhou, B. (2016). SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents. CoRR, abs/1611.04230. Opgehaal van http://arxiv.org/abs/1611.04230

[3] Rush, A. M., Chopra, S., & Weston, J. (2015). A Neural Attention Model for Abstractive Sentence Summarization. https://doi.org/10.18653/v1%2FD15-1044

[4] Sinha, A., Yadav, A., & Gahlot, A. (2018). Extractive Text Summarization using Neural Networks. CoRR, abs/1802.10137. Opgehaal van http://arxiv.org/abs/1802.10137

[5] Improving Neural Abstractive Document Summarization with Structural Regularization (Li et al., EMNLP 2018) https://aclanthology.org/D18-1441

[6] Lin, C.-Y. (2004, Julie). ROUGE: A Package for Automatic Evaluation of Summaries. Text Summarization Branches Out, 74–81.Opgehaal van https://www.aclweb.org/anthology/W04-1013

[7] CNN/DailyMail news dataset: https://www.kaggle.com/gowrishankarp/newspaper-text-summarization-cnn-dailymail

[8] Amazon Fine food Reviews: https://www.kaggle.com/snap/amazon-fine-food-reviews

[9]https://github.com/himanshurawat443/Abstractive-Text-Summarization-using-seq-2-seq-RNN-s/issues

[10]https://www.kaggle.com/code/singhabhiiitkgp/text-summarization-using-lstm/notebook

[11]https://towardsdatascience.com/text-summarization-using-deep-neural-networks-e7ee7521d804

[12]https://www.analyticsvidhya.com/blog/2019/06/comprehensive-guide-text-summarization-using-deep-learning-python/

[13]https://medium.com/hackernoon/text-summarization-using-keras-models-366b002408d9