International Journal of Hybrid Innovation Technologies
Vol.3, No.1 (2023), pp.29-40
http://dx.doi.org/10.21742/ijhit.2653-309X.2023.3.1.03

Comparing Abstractive and Extractive Approach for Text Summarization

Pinak Divecha[1] and Jinan Fiaidhi[2*]

[1,2*]*Department of Computer Science, Lakehead University, Canada*
[2*]*jfiaidhi@lakeheadu.ca*

Abstract

Researchers are interested in text summarizing because of its practical uses. We investigated and implemented both Abstractive and Extractive approaches to text summarization in this paper. These techniques have been used to summarize not only simple text but also general text and documents. This paper provides both a supervised and unsupervised technique to summarize, depending on the goal. Abstractive summarization is based on supervised learning, in which the text is interpreted and examined using advanced natural language techniques, and a new shorter text is generated that contains the most significant and helpful text from the original text. These summaries are more complex and perform similarly to summaries created by humans. We implemented both of the approaches in this paper based on their real-world application, as well as the summaries evaluation process. The approach compares the genuine human-written summary with the machine-generated summary, judging the quality of the summary based on the summary's significant terms and length. Overall, such a paper might be extremely beneficial to people in a variety of situations.

Keywords: *Text summarization, Abstractive, Extractive, Recurrent neural network*

1. Introduction

People have been devastated by the amount of knowledge available online in the form of media, journal and magazine articles, expert opinions, blogs, personal experiences, books, encyclopedias, and web pages in recent years due to the dramatic rise of the internet. Individuals will not be able to read all of the information and extract useful information from it. This minimizes the reading time and document selection process, so this type of situation requires the study of automatic text summarization. According to Ref et al. [1], a summary is defined as "The generated text which is formed from more than one text resource has comparatively vital information however it is not more than half of the authentic text(s) and usually, appreciably much less than that". News article summaries, maintaining a patient's medical history for future treatment, discovering important content in encyclopedias, collecting useful information from films, and information summaries for corporate or government officials are some real-world applications of automatic summarizations.

Article Info:
Received (May 14, 2023); Review Result (June 15, 2023); Accepted (July 30, 2023)

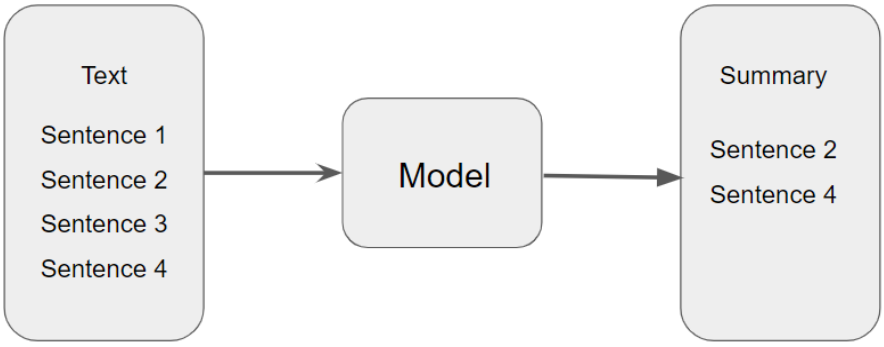

Figure 1. Extractive summarization

Extractive summarization and Abstractive summarization are the two types of text summarizing methodologies. Extractive text summarization [Figure 1] is a process that focuses on the important text or sentences within the text, then ranks the important sentences using various methods such as the word probability method, term frequency-inverse document frequency method, graph-based method, and machine learning based method, and generates summaries using the ranked sentence. As a result, when extractive summarization is used, each word or text in the summary belongs to the original text.

On the other hand, abstract summarization [Figure 2] is based on deep learning, in which completely new words and phrases are generated, separate from the original text, but the context of the sentences is kept, allowing us to interpret the resulting summary as human-made. To put it another way, this method uses powerful natural language processing algorithms to analyze text and provide a summary [2]. As a result, the abstract summary approach is substantially more difficult than the extractive summary approach. The extractive summary is shown to be more effective because it does not involve natural language creation or semantic representations.

There are primarily two ways to evaluate summaries: human review and machine evaluation. In human evaluation, human experts score the generated summaries based on how well they cover the main idea of the text, the answers to the queries, and grammatical errors. In the automatic evaluation, on the other hand, the review is carried out by a machine, which compares the abstractive or extractive summaries with the original human-generated summaries. Rouge score [3], BLEU (bilingual evaluation understudy) score, and F-score are examples of automatic evaluation.

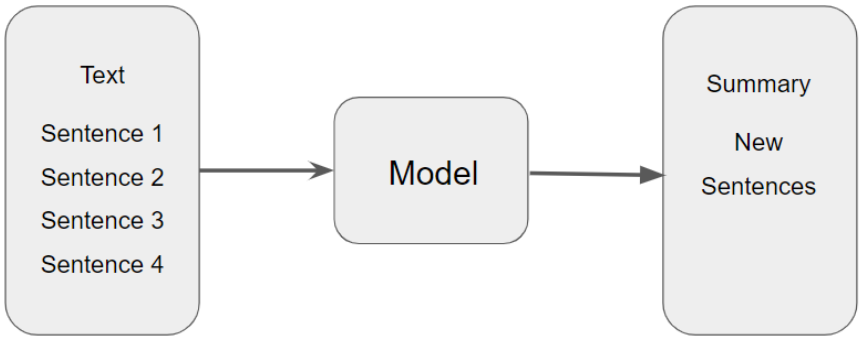

Figure 2. Abstractive summarization

2. Literature review

In recent years, the field of automatic text summarizing has gotten a lot of interest as a way to overcome the difficulties of producing decent summaries. In the field of extractive and abstract summarization, some substantial research work has been done, as described below. We employed the TextRank algorithm for extractive summarization, which was first published in the publication TextRank: Bringing Order into Texts [3]. We introduced the TextRank algorithm, a graph-based ranking model for processing text and using it for various applications in natural language processing, in this work. Our proposed method is an unsupervised learning method that requires no data training, making it highly adaptable to other domains, languages, and genres. The major reason this algorithm works so well is that it analyzes information recursively extracted from the entire text rather than just the local text. It employs an iterative mechanism to go beyond the simple graph mechanism and rank the phrases according to the text's relevance. The system represents sentences in an article with a weighted undirected graph and orders them using Google Page Rank [4]. They used the single document evaluation task on 567 articles from the Document Understanding Evaluations to test the text Rank algorithm [4]. They employed a technique called automated evaluation. The rough score and outcome for the 100-word single-document summary were around 0.48, indicating that the algorithm works effectively for single-document summaries [4].

We focused on the recurrent neural network (RNN) and long short-term memory (LSTM) architecture with an attention mechanism for abstract summarization. Deep Learning Based Abstractive Text Summarization: Approaches, Datasets, Evaluation Measures, and Challenges [2] was the subject of our review. We have detailed in this article the many datasets that are accessible for abstract summarization, as well as the various ways that are available to construct summaries utilizing the datasets after they have been generated. These dataset links are attached in Appendix A.

The numerous approaches available for evaluating the summary are briefly outlined in the appendix. There are various datasets available but we are using Amazon Fine Food Reviews and News Summary datasets whose links are attached in the appendix. Then explain how the RNN-LSTM model performs text summarization for both single-sentence and multi-sentence summarization, as well as what role the attention mechanism plays in natural language processing, as it was already used for neural machine translation, and how it aids in improving the results in text summarization tasks. Data preprocessing in the datasets helped me to improve these outcomes. All of the above-mentioned work such as RNN-LSTM, and page rank algorithm [5] which is an automatic text summarization method, was extremely beneficial in terms of gaining a foundational understanding of both approaches to the summary, which we then attempted to apply to certain text summarization applications. The extractive summarization with the Text Rank algorithm [6] is briefly explained in the next section, as well as how to utilize the method to generate summaries from text and Wikipedia webpages.

3. Methodology

We'll give you a quick overview of the architecture of the approaches I've applied in this part.

Pre-processing

The data preparation stage is necessary since the accuracy of the algorithm will be low if we utilize the raw data as input data. The first step we took with the raw data was to break it

down into individual sentences. After separating the phrases, we eliminated the stop words from each one since, as we saw previously when comparing sentences based on similarity, the algorithm favors stop words over other significant terms. After that, we eliminated any punctuation marks such as "the," "a," "an," "in," and so on, as well as looking for and removing duplicate terms. This is the most common method and is used by many researchers working in the natural language domain.

[Table 1] shows the preprocessing output on a few sentences which are used to train the model. In the table, you can see the sentences are converted into lower cases, which is followed by the removal of HTML tags, elimination of punctuation and special characters, and removal of stop words. The sentences obtained after the preprocessing are used for word embedding which is covered in the following section.

Table 1. Shows some examples of preprocessing

Input Sentences	Preprocessed Sentences
I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.	bought several vitality canned dog food products found good quality product looks like stew processed meat smells better labrador finicky appreciates product better
Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. Not sure if this was an error or if the vendor intended to represent the product as "Jumbo".	product arrived labeled jumbo salted peanuts peanuts actually small sized unsalted sure error vendor intended represent product jumbo
This is a confection that has been around a few centuries. It is a light, pillowy citrus gelatin with nuts - in this case Filberts. And it is cut into tiny squares and then liberally coated with powdered sugar. And it is a tiny mouthful of heaven. Not too chewy, and very flavorful. I highly recommend this yummy treat. If you are familiar with the story of C.S. Lewis' "The Lion, The Witch, and The Wardrobe" - this is the treat that seduces Edmund into selling out his Brother and Sisters to the Witch.	Confection around centuries light pillowy citrus gelatin nuts case filberts cut tiny squares liberally coated powdered sugar tiny mouthful heaven chewy flavorful highly recommend yummy treat familiar story lewis lion witch wardrobe treat seduces edmund selling brother sisters witch

GloVe for Unsupervised Learning [7]

As we all know, machines cannot grasp the English language or any other language in the same way that humans can, therefore we must utilize word embeddings to translate sentences into vectors, where each word has its vector that the machine can understand and operate with. We used GloVe [7] word embedding, which is one of many word embeddings available for converting words into vectors.

```
# Extract word vectors
word_embeddings = {}
f = open('/content/drive/MyDrive/NLP Project Final/glove.6B.100d.txt', encoding='utf-8')

for line in f:
    # print(line)
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    word_embeddings[word] = coefs
f.close()

sentence_vectors = []
for i in clean_sentences:
    if len(i) != 0:
        v = sum([word_embeddings.get(w, np.zeros((100,))) for w in i.split()]) / (len(i.split()) + 0.001)
        print(v)
    else:
        v = np.zeros((100,))
    sentence_vectors.append(v)

len(sentence_vectors)

print(sentence_vectors)
sim_mat = np.zeros([len(sentences), len(sentences)])
```

Figure 2. (b) GloVe implementation

GloVe [7] is a new global log-bilinear regression model for unsupervised word representation learning that outperforms previous models on tasks such as word analogies, word similarity, and named entity recognition. Following the conversion to vectors, the next step is to create a similarity matrix and identify the key sentences for a summary.

Here we have used the GloVe [7] word embedding as you can see in [Figure 2(b)]. To arrive at a consolidated vector for a sentence, we will first retrieve vectors (each with 100 elements) for the constituent words in the sentence, and then take the mean/average of those vectors [6].

Approach for Extractive Text Summarization

As previously stated, extractive summarization generates summaries by picking the most relevant words or sentences from the original text. The main goal of our suggested method is to provide a bullet-point summary of the text or document. We used the Text Rank algorithm [5] because there are several techniques for generating the extracted summary. The algorithm is called Text Rank because it functions similarly to Google's PageRank algorithm.

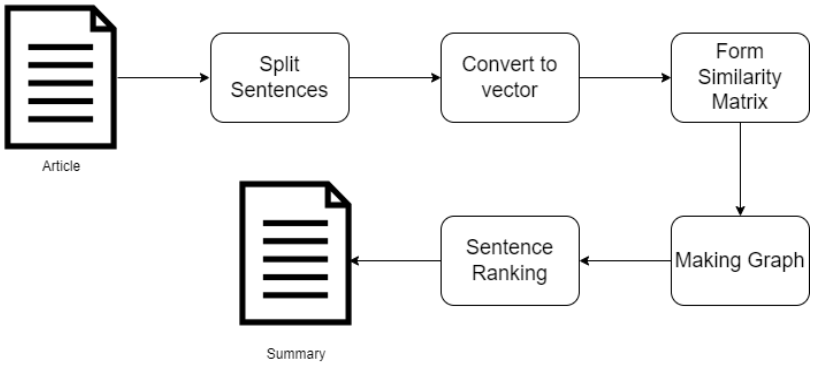

Figure 3. Extractive flow diagram

TextRank is an unsupervised method for automated natural language text summarization. It's a technique known as extractive summarization. Graph-based ranking algorithms use data derived from the graph structure to determine the value of a vertex in a graph. Text Rank is a natural language processing extractive summarization approach based on graphs. The Text Rank will be applied to the text created by the movies or Wikipedia pages. [Figure 3] depicts the algorithm's total operation. We'll start by extracting text from the article or videos and converting it to text. After that, each text will be broken down into sentences. We can readily interpret phrases in several languages as humans, but our machine only understands binary. As a result, we must turn the text into a vector, which requires the usage of word embeddings, which will be covered in depth later. We need to find the similarity between the phrases after transforming the text into vectors, and we used cosine similarity to do so. Graphs will be created from the generated similarity matrix. The important sentences will then be sorted using the PageRank algorithm, and the top-ranking sentences will be used as a summary.

Table 2 shows the ranking of the sentences calculated using the graph ranking algorithm. For the similarity, we have used the cosine similarity. Here, users can define the length of the summary, making sure that it should not exceed the total length of the input paragraph. The above ranking is done for the input sentence given in Table 3 below.

Table 2. Ranking of the sentences based on the cosine similarity

Rank of Sentence	Sentences
0.14817	During the course of his travels, he learns of his true purpose and meets many characters, including an "Alchemist", that teach him valuable lessons about achieving his dreams
0.1472	Inspired on learning this, he undertakes a journey to Egypt to discover the meaning of life and fulfill his destiny
0.1468	Towards the final arc, Santiago gets robbed by bandits who end up revealing that the "treasure" he was looking for is buried in the place where his journey began
0.1431	Santiago sets his sights on obtaining a certain kind of "treasure" for which he travels to Egypt

Table 3. Input paragraph

Santiago is a Shepherd who has a recurring dream which is supposedly prophetic. Inspired on learning this, he undertakes a journey to Egypt to discover the meaning of life and fulfill his destiny. During the course of his travels, he learns of his true purpose and meets many characters, including an "Alchemist", that teach him valuable lessons about achieving his dreams. Santiago sets his sights on obtaining a certain kind of "treasure" for which he travels to Egypt. The key message is, "when you want something, all the universe conspires in helping you to achieve it." Towards the final arc, Santiago gets robbed by bandits who end up revealing that the treasure" he was looking for is buried in the place where his journey began. The end.

Approach for Abstractive Text Summarization

The seq2seq model can be used to process and make the model understand sequential data. Traditional feed-forward neural networks take all available test instances into account separately. For example, if we use any individual stock as our sequential information, it is dependent on several factors such as the stock's opening value, volume, current price, and so on. The previous day's stock price is the most important factor in predicting the stock price. Such a dependency can be handled by a typical network. The recurrent neural network (RNN) is used in this case [Figure 4] [8].

RNNs can handle sequence inputs, but they struggle with longer sequences. RNNs are fantastic for short contexts, but we need our models to be able to grasp and recall the context behind the sequences in the same way that a human brain can. This is not doable with a simple RNN. A huge amount of irrelevant data can obstruct the flow of pertinent data to the point where it is required. In this case, a Recurrent Neural Network fails [8]. RNN only retains things for a short length of time, therefore it may be repeatable if we just need the knowledge for a short period, but once a large number of words are fed in, it becomes forgotten. It completely transforms the existing data by introducing a function to add new details.

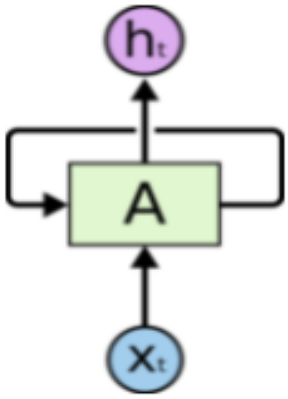

Figure 4. RNN has loops

As a result, the entire information is transformed, with no distinction made between 'essential' and 'non-essential' data. Long Short-Term Memory Networks are a modified variant of RNNs that can be used to overcome this problem [11]. LSTMs employ multiplications and additions to make tiny modifications to the data. LSTMs can determine whether the

information is remembered or forgotten when it passes via a process known as cell states. Each cell state has three dependencies based on the information available.

- The prior state of the cell (the available information in the memory after the previous time step);
- The previously concealed state (the generated output from the previous cell);
- The current time step's input (The new information that we will provide).

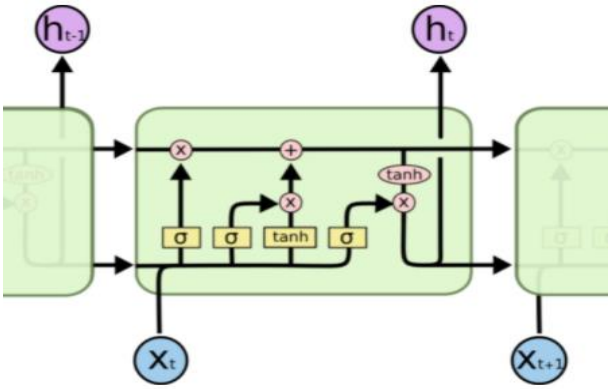

Figure 5. The repeating module in an LSTM contains four interacting layers in a figure

The LSTM architecture takes three inputs: X_t is the current time step input, h_t-1 is the last unit's output, and C_t-1 is the last unit's memory. The outputs h_t and C_t, on the other hand, correspond to the current output and memory of the same unit, respectively. As a result, when this single unit makes a decision, creates a new output, and alters its memory, it considers the current input, prior output, and previous memory. In LSTM, two gates can be used to accomplish this task:

- Memory checkpoint
- Forget about the gate

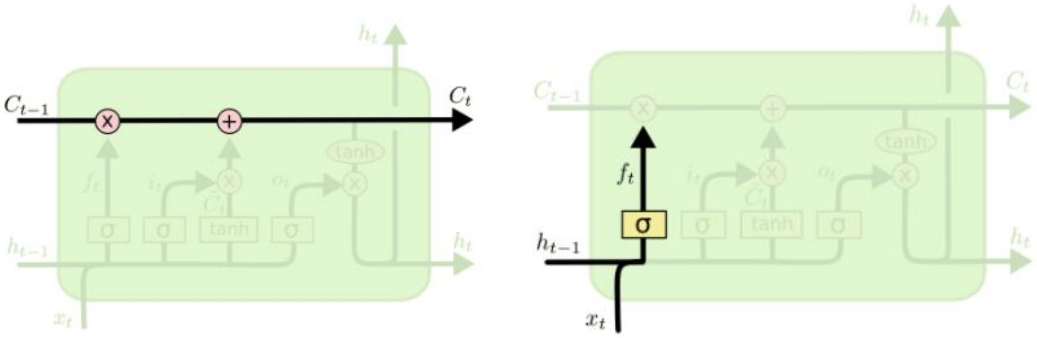

Figure 6. Input gate layer

The flow of the memory control gate work is depicted in [Figure 6] in the highlighted area. It accepts data from the old C_t-1 memory. The forget gate, which performs element-wise multiplication, is the first "X." Multiplying the old memory C_t-1 with a vector close to 0

indicates that the majority of the old memory should be forgotten. The "+" operator, which is a piece-wise summation, is the next operation. It will bring together old and new memories.

The "X" operator below the "+" sign, on the other hand, is in charge of determining how many portions of fresh memory will be merged into the existing memory and then updating the memory as new memory C_t becomes available.

Figure 6(b) shows a single-layer neural network with the following inputs: h_t-1, the previous LSTM block's output, X_t, the current LSTM block's input, C_t-1, the previous block's memory, and finally a bias vector b_0. This neural network employs a sigmoid activation mechanism to generate the forgotten valve as an output vector, which can then be added to the old memory C t-1 via element-wise multiplication.

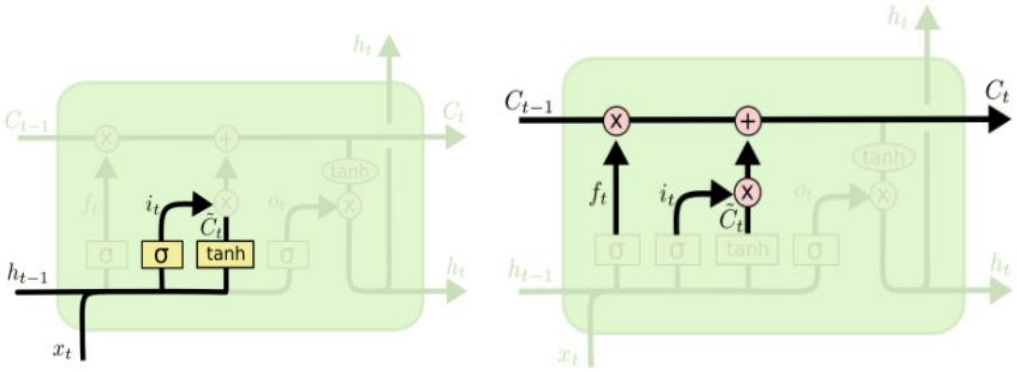

Figure 7 (a): This step helps to decide what new information we need to store in the cell

The single-layer neural network in [Figure 7(a)] acts as a new memory gate, accepting the same inputs as the forget gate except for the bias vector. It is in charge of regulating the amount of influence a new memory can have on an older memory. A different neural network, on the other hand, is responsible for the creation of new memory [14]. It's a simple one-layer network with tanh activation. The output of the new memory gate and this layer will be elementally multiplied before being combined with the old memory. C_t represents the overall result of this operation, as seen in [Figure 7(b)].

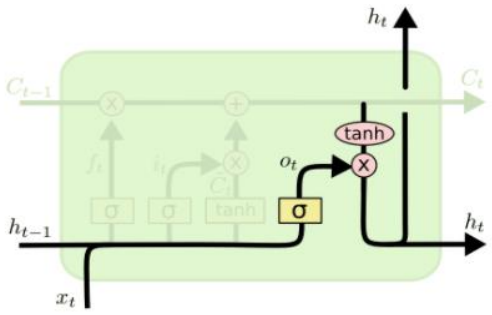

Figure 7 (b). Output layer

This is the last phase in the LSTM block's output generation process, as shown in [Figure 8]. To generate output, this phase uses a bias vector, a fresh memory, the previous output h_t-

1, and the input X _t as inputs. The quantity of new memory that is delivered to the following LSTM block is controlled by this valve.

We have implemented this LSTM method in my model and ran the model for a variety of epochs but for 14 epochs the loss reaches a minimum of 2.3 and for the test cases the model can generate the most effective sentences for the abstractive approach when we have used the News Summary Dataset.

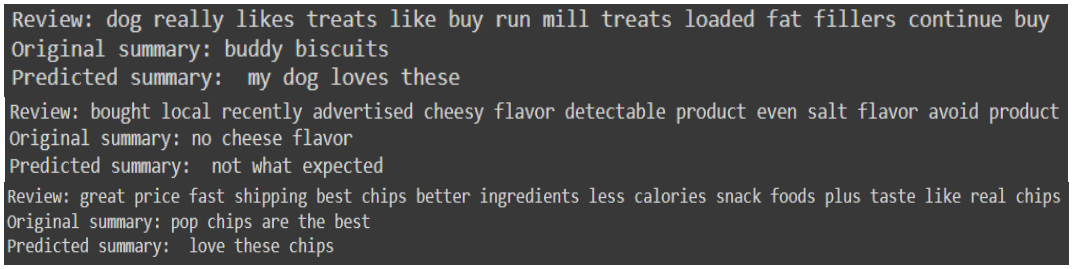

Figure 8: Results of abstractive text summarization

[Figure 8] shows the output result of the abstractive text summarization model. Here the original summary is the summary generated by humans while the predicted summary is the summary generated by our model where we have used the LSTM and attention layer.

Performance

Working on both the method i.e. abstractive and extractive methods, the extractive method is unsupervised learning and it doesn't require any model to be trained. Thus, the extractive method is pretty fast in execution. Also, it generates the summary from the input sentences. On the other hand in an abstractive method, we have trained the model on the two different datasets [12][13]. Here the news summary [12] dataset is a multiline article dataset while the Amazon review [13] dataset is a single-line article dataset. Our model generates better outcomes in single-line datasets compared to the multiline articles. Also, in terms of training the model, our model takes 2-3 days to get trained on GPU.

4. Conclusion

The researchers described and shown the approaches we used for automatic text summarizing for news article datasets, Wikipedia pages, any textual content, and text documents in this work. The researchers looked at both the extractive and abstract summarizing methodologies. And also used the attention mechanism with recurrent neural network models that use LSTM encoders and decoders. The proposed abstractive model works well for single-line documents compared to multiline documents. The model gives the best output after 14 epochs as the loss reaches a minimum value of 2.3. With this hyperparameter, the model can regenerate better results compared to another number of epochs (5,6,10,12). Here we can't compare the two approaches as the attractive model generates a new summary based on the input while the extractive method finds the rank of the input sentences and lists the sentences based on rank for a summary.

5. Future Work

The researchers like to work on a hybrid strategy for text summarizing in the future after implementing both strategies to increase the quality of the machine-generated summary.

While generating a summary, such a method will consider the most significant term and keep it in the same form, perhaps improving the summary's quality. With the Django framework, the researchers would like to develop an application. In that application, it will use the model built using the hybrid strategy and summarize the audio, and Wikipedia summaries.

Acknowledgment

We wanted to deeply thank Dr. Jinan Fiaidhi, who took a keen interest in my paper and helped us all along, till the completion of our paperwork by providing all the necessary information, guidance, and suggestions for achieving such good results. We would like to express our gratitude towards our classmates who suggested some ideas to improve my paper results.

Appendix A

There are three different files which are Jupyter notebook files. To run the Jupyter Notebook, install Anaconda as it provides all the requirements to run the Juypter Notebook. To install the Anaconda installer for Windows kindly go through the steps shown in this link: https://docs.anaconda.com/anaconda/install/windows/. To run the Python file as well as the jupyter notebook, the following libraries will be required, among them many libraries are pre-installed with Anaconda.

- pandas: Pre installed with anaconda.
- NumPy: Pre-installed with Anaconda.
- random: Pre-installed with anaconda.
- matplotlib: Pre-installed with anaconda.
- sklearn: conda install scikit-learn
- Tensorflow: pip install tensorflow==2.4

Here, to run the code, I'm using a specific version of TensorFlow which is 2.4. The jupyter notebook contains the importing of data, pre-processing of data, defining the model, and running the model on a test case.

References

[1] D. R. Radev, E. Hovy, & K. McKeown, “Introduction to the special issue on summarization,” Computational linguistics, vol.28, no.4, pp.399-408, **(2002)**

[2] D. Suleiman and A. Awajan, “Deep learning based abstractive text summarization: Approaches, datasets, evaluation measures, and challenges,” Mathematical Problems in Engineering, (2020) DOI: 10.1155/2020/9365340

[3] A. Pai, “Comprehensive guide to text summarization using deep learning in Python,” June 10, 2019, analyticsvidhya.com

[4] R. Mihalcea and P. Tarau, P. TextRank: Bringing Order into Texts, **(2004)**

[5] M. V. Balipa, H. Jathanna, C. Ramasamy, and Balasubramani, “Text summarization for psoriasis of text extracted from online health forums using text rank algorithm,” International Journal of Engineering and Technology, **(2018)**

[6] S. G. Tanwi, V. Kumar, Y. S. Jain, and B. Avinash, “Automatic text summarization using text rank,” International Research Journal of Engineering and Technology (IRJET), **(2018)**

[7] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” EMNLP, 14, 15321543, (2014) DOI: 10.3115/v1/D14-1162

[8] P. Srivastava, “Essentials of deep learning: Introduction to long short term memory,” (2017) analyticsvidhya.com
[9] S. Yan, “Understanding LSTM and its diagrams,” (2017) medium.com
[10] F. Shaikh, “Essentials of deep learning sequence to sequence modeling with attention (using Python),” (2018) analyticsvidhya.com
[11] A. Pai, “Comprehensive guide to text summarization using deep learning in python,” (2019) analyticsvidhya.com
[12] https://www.kaggle.com/datasets/sunnysai12345/news-summary?select=news_summary_more.csv
[13] https://www.kaggle.com/datasets/snap/amazon-fine-food-reviews