International Journal of Scientific Research in Science, Engineering and Technology

Print ISSN - 2395-1990
Online ISSN : 2394-4099

OPEN ACCESS

Available Online at : www.ijsrset.com
doi : https://doi.org/10.32628/IJSRSET12310669

Automated Text Summarization as A Service

[1]Ketan Shahapure, [2]Samit Shivadekar, [3]Shivam Vibhute, [4]Milton Halem
[12]Department of CSEE, University of Maryland Baltimore County, USA
[3]San Jose State University, San Jose, California, United States
[4]University of Maryland, Baltimore County

ARTICLEINFO

Article History :
Accepted: 01 Jan 2024
Published: 10 Jan 2024

Publication Issue :
Volume 11, Issue 1
January-February-2024
Page Number :
54-65

ABSTRACT

Recent advancements in technology have enabled the storage of voluminous data. As this data is abundant, there is a need to create summaries that would capture the relevant details of the original source. Since manual summarization is a very taxing process, researchers have been actively trying to automate this process using modern computers that could try to comprehend and generate natural human language. Automated text summarization has been one of the most researched areas in the realm of Natural Language Processing (NLP). Extractive and abstractive summarization are two of the most commonly used techniques for generating summaries. In this study, we present a new methodology that takes the aforementioned summarization techniques into consideration and based on the input, generates a summary that is seemingly better than that generated using a single approach. Further, we have made an attempt to provide this methodology as a service that is deployed on the internet and is remotely accessible from anywhere. This service provided is scalable, fully responsive, and configurable. Next, we also discuss the evaluation process through which we came up with the best model out of many candidate models. Lastly, we conclude by discussing the inferences that we gained out of this study and provide a brief insight into future directions that we could explore.

Keywords - Machine Learning, Natural Language Processing, Automated Text Summarization, Language Modeling

I. INTRODUCTION

In the 21st century, with the advent of the internet, there has been an explosion of information all across the globe. Each year, an increase in the consumption of digital media is seen. This data is highly unstructured and it a reader is expected to search and skim across the text in order to gain desired information. This, in turn, arouses the need for the summarization of longer pieces of texts. While manual summarization is possible, it is often time-consuming and requires a significant amount of effort from the

user's side. This makes it imperative to apply techniques that can summarize a given piece of text in an efficient and coherent manner. Text summarization is the process of generating concise summaries from a given input text.

This process tries to generate a summary by retaining all the crucial details, so as to not change the meaning of the original text. Furthermore, this helps in better comprehending the context. Automated text summarization, which can be done without human intervention, is an effective way to summarize a given piece of text in a fast manner. This can be done with the help of natural language processing, which is concerned with the interactions between human language and computers. But, there are certain difficulties concerning automated text summarization and the main challenge is to select crucial information from the given input text.

Text summarization [9] can be done in two ways, Extractive and Abstractive. In Extractive summarization, a summary of the given text is generated by considering the subset of the input text. No new words are introduced but the existing sentences are reordered. In this type of summarization, the most important sentences would be picked based on some metrics to generate the output summary. For abstractive summarization [14], the sentences are reinterpreted to generate new sentences. This approach generates a summary by understanding the original text using linguistic methods. This technique is considered to generate a more efficient summary as the sentences are reinterpreted rather than just reordered.

Language Models are used to summarize the input text in abstractive summarization. The transformers [18] is one of the majorly used libraries in the field of natural language processing. They [19] aim to solve the variable-length problem by using an encoder-decoder stack.

II. PROBLEM STATEMENT

We aimed to provide a service that creates cogent and relevant summaries. Further, as an extension, we intended to provide the user with more than one way to provide input, such as through URLs of publicly available websites. We also wanted to be sure that the language model we were using for summarization outperformed other models, and hence, we used the schemes of qualitative and quantitative assessments to evaluate multiple Transformers models. Our service aims to keep the users interested when they intend to fetch the girth of a large or short text in less time. For example, in the case of media and publishers, the ability to automatically provide summaries of all their content allows readers and visitors to focus on the information that interests them most, hence increasing the quality of service and engagement. Similarly, analysts and researchers can make use of this service to get a condensed form of their respective domains.

In the further sections, we present the existing techniques that were studied as the foundations and then present the methodology that we propose, then explain the service-oriented architecture of our application, and also touch upon the model evaluation process that was performed. As our end goal, we deployed an application that would generate efficient summaries by using combining abstractive and extractive methodologies.

III. RELATED WORK

There are different types of summarization techniques such as indicative, abstractive, extractive, generic etc. [21]. Out of these approaches, we mainly focused on abstractive and extractive ones.

A. Extractive Summarization

The extractive summarization approach picks up the top most important sentences from the input text and re-arranges them to generate the output summary. The priority of picking the sentences is majorly based on the linguistic features of the sentences, as explained by N. Moratanch et. al [20]. In the case of extractive summarization, every sentence in the output summary is a part of the input text. So, in this case, every sentence and word of the summary actually belongs to the original document which is being summarized. For generating the extractive summary of a given input text, there were multiple services available like Gensim [24], Spacy, and NLTK, out of which we have used Spacy [13]. Spacy [6] is an open-source NLP library that provides features like tokenization, text classification, and Rule-based matching. Spacy uses certain scoring metrics to rank the sentences and picks the top k sentences to generate the output summary.

B. Abstractive Summarization

Abstractive summarization is a technique that generates novel sentences by rephrasing or using new words, instead of simply extracting the important sentences [11]. Here, a summary is created by analyzing the semantic information about the text and with deep analysis and reasoning, and new sentences are generated from the original text. Following are some of the architectures studied for creating abstractive summaries:

1) Recurrent Neural Networks: A recurrent neural network (RNN) [26] is a type of artificial neural network where the connections between nodes create a cycle, and this allows the output of the node to form an input to the same node. According to Sun et. al. [29] RNNs are usually used to play two roles: as an encoder that transforms sequential data into vectors, and as a decoder that transforms the encoded vectors into sequential output. This method can generate sentences that are grammatically correct but can have issues generating longer sentences. [8].

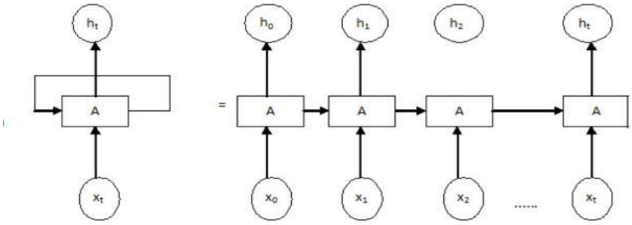

Fig. 1. Example of a Recurrent neural network [4]

2) Long Short Term Memory: Long Short Term Memory networks [28], or in short, LSTM, is a special kind of RNN that is considered more effective for learning long-term dependencies. This characteristic is useful, especially in sequence prediction problems. LSTMs [12] have feedback connections, which makes them capable to process the entire sequence of data, without forgetting the past states. Figure 2 represents the architecture of an LSTM.

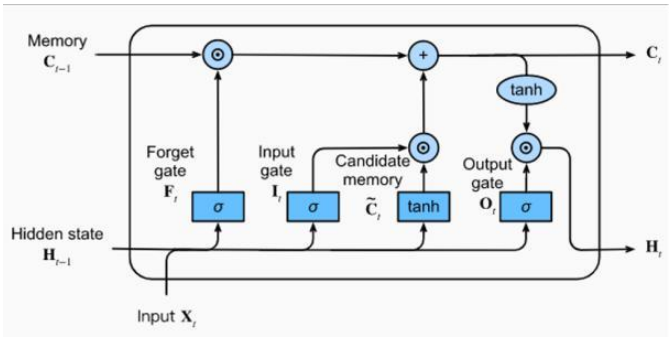

Fig. 2. Long Short term memory architecture [3]

With LSTM networks, the issue of vanishing gradients can be overcome. Although this method can help generate sentences that are grammatically correct, it can have issues generating sentences that are longer in length.

3) Transformers: Transformers [31] uses an encoder-decoder architecture to perform text summarization. According to Vaswani et. al., [30] an encoder will convert the variable-length input to a fixed length. Then, the hidden state goes through a decoder, which converts the fixed length to a variable length output. This behavior is required in complex problem domains

like machine translation, speech recognition, sequence prediction, etc. Figure 3 showcases the architecture of a transformer. With transformers, the relationship between sequential elements that are far from each other can be under-stood. They can train more data in less amount of time. All the words in the sentence get equal attention. In transformers, the sentence is processed as a whole, rather than word-to-word. The LSTM requires 8 steps to process a sentence while BERT 1] takes only 2.

IV. PROPOSED METHODOLOGY

After studying the architectures of extractive and abstractive approaches, it was concluded that there was no single approach that could cater to all summarization tasks as each of the aforementioned approaches had its own pros and cons. In extractive summarization, the sentences are merely shuffled around depending on their respective term weights. However, in this approach, semantics often take a backseat as the ranking of terms is done at each sentence level and so, the context of the entire text is not taken into consideration.

The problem of context not being considered is solved by the abstractive summarization approach wherein the language model first understands the entire piece of text using the encoder. A major advantage of the attention mechanism is that the model actually understands the input and tries to predict the tokens in its vocabulary with the highest probabilities with respect to those of the input logits. However, this might be a downside in some cases in which a user might not be comfortable with new words getting formed. This can be attributed to the fact that the predicted tokens might sometimes be out of context, or simply gibberish. Regardless, both these methodologies served as a basis to design an approach that would, somehow, be able to combine the pros of both these approaches and come up with a summary that is seemingly better than that which is created by either of these approaches.

Our initial approach to resolve this problem was to actually combine both approaches, that is, feed the input text to the Ex-tractive summarizer first, and then, feed the output generated to the abstractive summarizer. Although the approach seemed to work well in some cases, it did not generalize well for some larger pieces of text. Here, we came to know a limitation of the Transformers. Depending on the model architecture, the transformers only support tokens to a certain limit. After this limit has been reached, they simply ignore the rest of the tokens in the input.

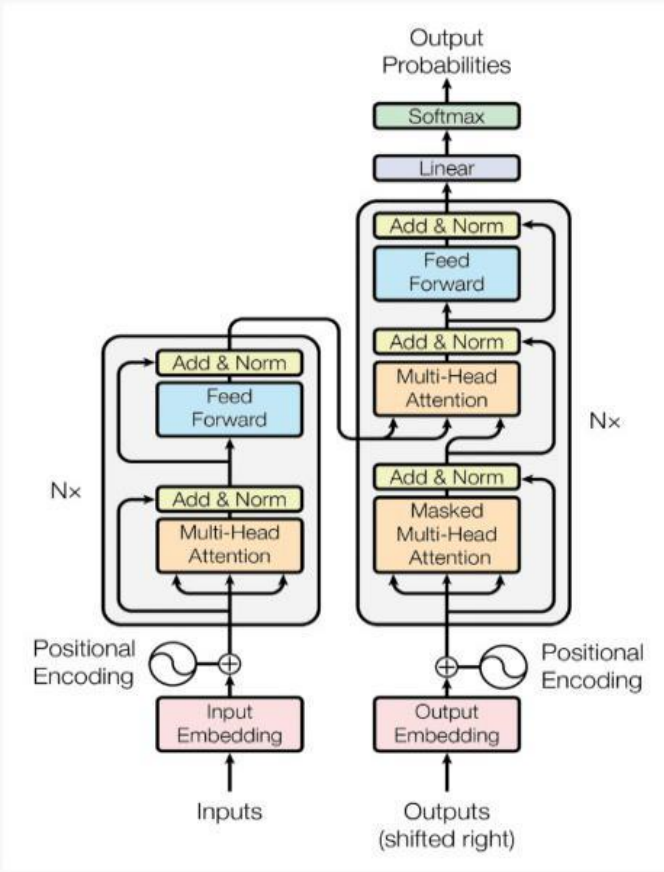

Fig. 3. Transformer architecture [30]

To solve this limitation, a chunking mechanism was intro-duced which maintained a collection of sentences based on their length of tokens. Each sentence was added to the collection and a counter was maintained to count the total number of tokens. If for a sentence, the count of its tokens exceeded the limit, it was stored in a new collection. The Transformers model then summarized the collections iteratively and

merged their summaries at the end while returning the output response. This, we found, yielded a noticeably higher quality of output.

Since this framework was to be deployed as a service, a third scenario was added in the final phase of product deployment wherein, the user was given a choice to select the type of summarization that he would prefer to be used. The subsequent section explains the practical implementation of the discussed methodology, although more from a service-oriented perspective.

V. IMPLEMENTATION

Based on the proposed methodology IV, an algorithm was designed to take in the user input, decide the type of summarization, and finally, return the summarization response. For extractive summarization, SpaCy [13] model seemed to provide far better results as compared to its counterparts. For abstractive summarization, HuggingFace's transformers [32] were used, but this selection was far more complicated than the former one. Since the HuggingFace's repository had more than a thousand models available, it was practically impossible to compare all these models. So, only a few of the models were tested. The results of these models can be found in section VI.

Since there was no way to evaluate the summaries that were to be generated at the run-time, we had to come up with a good model that would generate good outcomes in a real-world scenario. For this, we have used a qualitative and a quantitative approach to come up with the best model, and this model is now being used inside the application. Details of this model evaluation process are explained in the section VI.

A. Algorithm

The following steps represent the core ML algorithm of the proposed approach:

vi Take the input from the stakeholder. ii If the input is in the form of a URL:

- Pass the URL to the web scrapper
- The web scrapper crawls up the URL, retrieves the text from the website, and invokes the backend API.

vii If the input is in a text-based format, the API is invoked directly without the web parser coming into play.

viii Here, a sentence tokenizer is used to maintain a count of total sentences.

ix In case of default parameters:

x If there are less number of sentences, the extractive ap-proach is chosen, otherwise, the abstractive approach is picked for text summarization.

xi In case of custom parameters:

- Use the type of approach based on the user's choice and the expected length of the summary.

xii As the final step, return the output response from the summarization model to the user interface.

The steps mentioned above are elaborated further in technical depth in the following subsection.

B. Architecture

As the application was to be deployed as a service, service-oriented constructs were required to launch the application onto a production environment. Figure 4 represents the architecture of the deployed service model.

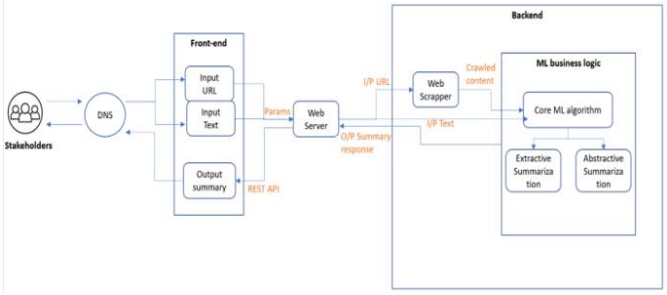

Fig. 4. Architecture of the deployed service

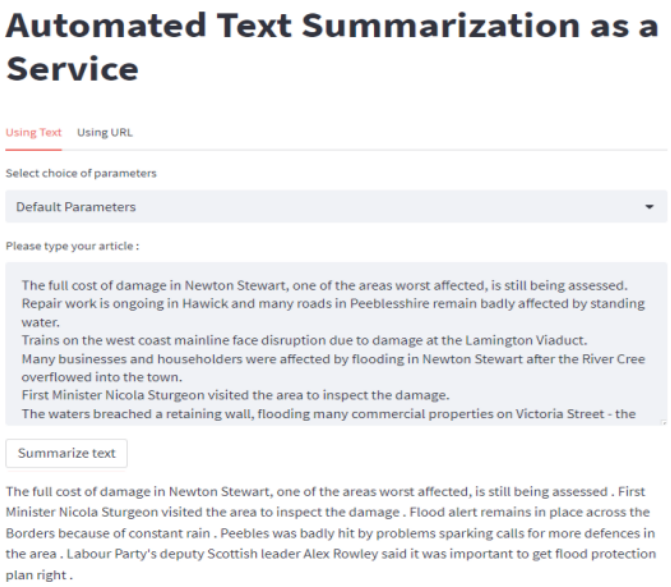

Fig. 5. UI of deployed service

In the deployed service, the workflow starts with the stake-holder. Here, it is assumed that the stakeholder already has the access to the text that he intends to provide to the application. Then, he is able to access the application through a public URL, which is accessible through a Domain Name System (DNS). This URL is provisioned by a public cloud named Streamlit [5].

1) Front-end: Streamlit [16] provides an interface to create quick and interactive and fully responsive user interfaces with-out relying on technologies such as HTML, CSS & JavaScript. After accessing the URL, the user is able to provide the textual input through the user interface. Here, a choice is offered to the user to provide the input either as actual text or the URL of the web page that he intends to summarize. Now, there can be two forms of inputs:

- Text-based input:

If the user has the actual text, he can copy the entire text and paste it into the text box provided in the user interface. In this type, the input is directly passed over to the backend business logic.

- URL-based input:

In the later phases of product development, it was noticed that it often became a tedious task to manually copy the content when the input text was hosted on a public website. So, a new micro-service was designed as a feature to overcome this challenge which used the concept of 'web-crawling' [25].

2) Back-end: After the user provides the input, the web server invokes the API of the backend business logic. The core ML logic then, depending on the type of input dynamically decides the type of summarization method to be used. After extensive experimentation, it was surmised that inputs that had less number of tokens were more suited for the extractive approach, and those having a larger number of tokens were passed to its counterpart.

In the later stages of production, a new feature of parameter configuration was added the service wherein the user was provided the option to override the default parameters and explicitly choose the type of summarization if at all he intended to.

3) Deployment: The aforementioned application is deployed as a service using Streamlit Cloud [5].

Figure 5 presents the user interface that the stakeholders can access to provide the input text and obtain the summaries.

As this service is provided for the end-users, we provide the user with the option to override the default configurations if he intends to choose any type of summarization approach. Now, ideally, for longer texts, an abstractive approach is set to be selected. But, if a user is aware of these approaches and explicitly wants to select the extractive method, Figure 6 showcases the selection of that.

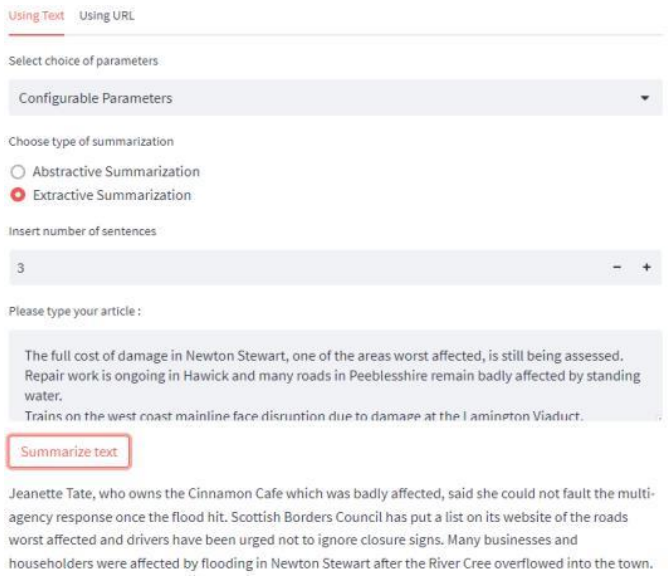

Fig. 6. Manually selecting extractive summarization approach

These features are intended to provide instructiveness, flexibility, and convenience.

VI. VI. MODEL EVALUATION

For the extractive part, the SpaCy model seemed to give quite better results during the initial phases of development. Hence, not many alternatives were explored in place of SpaCy. However, as mentioned before, choosing an abstractive model was quite complex. To test out different models in the Hugging Face repository [2], we tried different pre-trained models, which are as follows:

BART model is a Transformer's sequence-to-sequence model, and the Bart-Large variant of the model - developed

by Facebook - as described in [17] was selected as one of the models. Another model that was tried out was the distilled version of BART, known as DistilBart [27], which according to HuggingFace, was well-known for text generation and summarization use cases. This model is trained on the news summarization dataset of CNN [22]. T5, likewise, [23] was another common text-to-text Transformer model used that was trained in English & was suitable for NLP tasks. We also made use of the mT5 multilingual XLSum model [33], which is a multilingual revised version of the T5 model and trained on 45 languages with XL dataset [7]. Lastly, we used the RuBERT [15] model, which was pre-trained with additional languages of Roman Urdu.

Initially, we visually inspected the summaries generated by different language models. But, since summarization can be a subjective process, a quantifiable measure was needed alongside a qualitative assessment to evaluate the model's performance.

A. Quantitative Assessment

For quantitative analysis, ROUGE scores [10] were used to compare human and machine-generated summaries. These scores were based on the summaries that worked by checking the overlapping n-grams. Further, each model's runtime was measured in seconds to get an idea about the time taken by each model to generate output. This was repeated for 100 summaries from the XSUM dataset and the results were aggregated.

TABLE I. ROUGE SCORE

Model name	R-1 Score	R-2 Score	R-L Score
Distilbart-cnn-12-6	0.29	0.12	0.20
Bart-large	0.21	0.08	0.12
mT5 multilingual XLSum	0.20	0.03	0.08
Unifiedqa-t5-small	0.12	0.07	0.15
T5-large	0.24	0.04	0.09
Rubert telegram headlines	0.09	0.02	0.14

TABLE II. EXECUTION TIME

Model name	Performance Time (in seconds)
Distilbart-cnn-12-6	8.35
Bart-large	23.10
mT5 multilingual XLSum	10.40
Unifiedqa-t5-small	11.65
T5-large	19.65
Rubert telegram headlines	9.40

In the real world, as there are no human-generated summaries available for each and every piece of text, this assessment was crucial to have a good

understanding of how the model would perform in real-time. These scores are not a part of the service and were mainly used to select the optimal transformers model to be used inside the production environment.

B. Qualitative Assessment

For qualitative assessment, analyzing the summaries generated by each model was necessary. This part of the experiment was to actually read the entire original text followed by summaries of each model. So, all the input texts and the generated summaries were read to get an idea about the visual aspects of the model's outputs. This helped in identifying the anomalies that were present in the form of bad grammar, odd punctuation, irrelevant characters, and repeating tokens. Following is an example of the actual text that was used for evaluation:

The full cost of damage in Newton Stewart, one of the areas worst affected, is still being assessed. Repair work is ongoing in Hawick and many roads in Peeblesshire remain badly affected by standing water. Trains on the west coast mainline face disruption due to damage at the Lamington Viaduct. Many businesses and householders were affected by flooding in Newton Stewart after the River Cree overflowed into the town. First Minister Nicola Sturgeon visited the area to inspect the damage. The waters breached a retaining wall, flooding many commercial properties on Victoria Street - the main shopping thoroughfare. Jeanette Tate, who owns the Cinnamon Cafe which was badly affected, said she could not fault the multi-agency response once the flood hit. However, she said more preventative work could have been carried out to ensure the retaining wall did not fail. Meanwhile, a flood alert remains in place across the Borders because of the constant rain. Peebles was badly hit by problems, sparking calls to introduce more defenses in the area. Scottish Borders Council has put a list on its website of the roads worst affected and drivers have been urged not to ignore closure signs. The Labor Party's deputy Scottish leader Alex Rowley was in Hawick on Monday to see the situation first hand. He said it was important to get the flood protection plan right but backed calls to speed up the process.

Below are the results of each model's summary:

1) DistilBart:

First Minister Nicola Sturgeon visited the area to inspect the damage. Many roads in Peeblesshire remain badly affected by standing water. Flood alert remains in place across the Borders because of constant rain. Labor Party's deputy Scottish leader Alex Rowley was in Hawick on Monday to see the situation first hand. He said it was important that "immediate steps" were taken to protect areas most vulnerable and a clear timetable put in place for flood prevention plans.

2) BartLarge:

Many businesses and householders were affected by flooding in Newton Stewart after the River Cree over-flowed into the town. The waters breached a retaining wall, flooding many commercial properties on Victoria Street - the main shopping thoroughfare. Peebles was badly hit by problems, sparking calls to introduce more defenses in the area. Repair work is ongoing in Hawick and many roads in Peeblesshire remain badly affected by standing water. Many businesses and householders were affected by flooding in Newton Stewart after

3) mT5 multilingual XLSum:

Residents in Dumfries and Galloway have said they are "neglected or forgotten" after flash flooding caused major disruption to homes, railways and train services on the Scottish Borders were badly affected by Storm Ciara last week. BBC News has been told there is no clear timetable of how it was handled as soon as next month. (Here'd be some other things about what happened with our lives).

4) Unifiedqa t5 small:

No. we have been affected by flooding in dumfries and the nith, but are still being assessed for it now? yes! there is an immediate response from all over the town to see how much damage has suffered on their property. they are mostly victims of flooding at home or abroad so this will help them avoid being arrested as soon as-you go through with the flood protection plan (for example) because people are vulnerable towards more than 11,000 per year while others are homeless

5) T5 Large:

A clean-up operation is under way in parts of Dumfries and Galloway which were hit by flooding over the weekend. ”Obviously it is heart-breaking for people who have been forced out of their homes and the impact on businesses.” He said it was important that ”immediate steps” were taken to protect the areas most vulnerable and a clear timetable put in place for flood prevention plans. The railway line between Edinburgh and Locker-bie remains closed due to damage caused by Storm Desmond’s heavy rain last week.

6) Rubert telegram headlines:
The Royal Variety of Defense (): fashion off the highline to home for North Tatara, Lord NNC) #kulturlyHess (Deatic Ukraine”. [] Foods’ on myke Home Offensiveboard-Store — 2020/RUT) ’5 % % ff*********)) Dow Jones!

C. Observations
This section of the paper describes the overall observations from the perspective of a human. Following are the subtleties and intricacies observed in each model’s performance:
1) Distilbart:

The summary of Distilbart seems to be one of the most comprehensible summaries of all the output summaries. Along with a good ROUGE score, Distilbart also manages to bring a cohesive summary within good performance time as indicated in Table 2.

2) BartLarge:

The Bart Large model provided a lucid and legible summary. However, it also represents poor punctuation. Unlike the remaining models, BartLarge provided a pre-cise version of the input text, but it comes along with a cost of high-performance time of 23.1 seconds. This will hinder the efficiency of this service for its large execution time. Also, there were tokens that were seen to be repeating.

3) mT5 multilingual XLSum:

Although the mT5 multilingual XLSum model provided grammatically correct paragraphs in the form of summaries, it also brought irrelevant context with itself. The additional text sometimes reduced the essence of summarized output and made it more trivial. This part aligned with the model’s low ROUGE score as represented in Table 1.

4) Unifiedqa t5 small:

The model Unifiedqa t5 small reflects two major drawbacks, one including the poor grammar followed by the extraneous questions put forward in the output summary. The girth of the condensed output was not pertinent to the matter under consideration.

5) T5 large:

Unlike other models, except Distilbart, the T5 large model’s shortened text as the output summary was better with respect to clarity and explanation. Regardless of how clearly the passage reflected the meaning, the evident downside of the model includes its high execution time which makes the summaries

too long to generate accompanying poor capitalization as well.

6) Rubert Telegram Headlines:

This RuBERT model brings forward the most redundant and superfluous matter in the output summary among all the tested models. Along with the extraneous characters present in the summary, the model failed to bring concise, generic, and coherent content.

VII. INFERENCES

The results obtained from the assessments were aggregated and analyzed. Interestingly, the observations of qualitative analysis aligned with the quantitative ones. When it came to picking the best model out of the 6 models, DistilBart seemed to outperform other models in terms of ROUGE metrics, performance time and more importantly, the quality of the summary. The model provided the most comprehensible summary among all the six models used, and hence, is now being used in the application.

VIII. LIMITATIONS & FURTHER DIRECTIONS

Currently, we are using a tokenizer to convert the input text into chunks of data. This tokenizer has a specific size of vocabulary available. As part of the future scope, the tokenizer can be extended to include more words and thus, enrich its vocabulary.

Secondly, for this deployment, we have used the distilled version of the BART model which is trained on a news dataset. In the future, we can 'retrain' the model on different data sets to better generalize the performance of the model. This could provide summaries for input texts if the model is trained on different datasets apart from the one trained on CNN news.

One other limitation of the service is the input format. Currently, the service that we have deployed accepts only two forms of inputs - text-based and URL-based. In the future,we can increase the forms of inputs and thus, provide the user with more options to pass the input text, such as PDF, DOCX, TXT, and so on. This would enable the user to possess more flexibility and would also allow for better convenience while using the service.

IX. CONCLUSION

In this paper, a brief study of the different techniques for text summarization was presented. Apart from that, a new method of leveraging the summarization models was presented, also discussing its effects on the summaries and how it could help summarize better. Further, the architecture of the built service was discussed and the core algorithm of the application was explained. A comparative evaluation of different language models was then made through qualitative and numeric means and the results were put forth. From the results, it was concluded that for the tasks which required coherence, DistilBART seemed to be the optimal choice. This model is now being used inside the deployed service.

X. REFERENCES

[1]. Bidirectional attentional encoder-decoder model and bidirectional beam search for abstractive summarization. https://intellipaat.com/blog/what-is-lstm/.
[2]. Hugging face. https://huggingface.co/models.
[3]. Lstm architecture. https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c.
[4]. Rnn architecture. https://www.researchgate.net/figure/An-unrolled-Recurrent-Neural-Networkf ig2321811462.
[5]. Streamlit cloud. https://streamlit.io/cloud.

[6]. Text summarization with spacy. https://spacy.io/usage/spacy-101.

[7]. Xlsum. https://huggingface.co/csebuetnlp/mT5multilingualX LSum.

[8]. Kamal Al-Sabahi, Zhang Zuping, and Yang Kang. Bidirectional attentional encoder-decoder model and bidirectional beam search for abstractive summarization. arXiv preprint arXiv:1809.06662, 2018.

[9]. Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Eliz-abeth D Trippe, Juan B Gutierrez, and Krys Kochut. Text summarization techniques: a brief survey. arXiv preprint arXiv:1707.02268, 2017.

[10]. Kavita Ganesan. ROUGE 2.0: Updated and improved measures for evaluation of summarization tasks. CoRR, abs/1803.01937, 2018.

[11]. Som Gupta and Sanjai Kumar Gupta. Abstractive summarization: An overview of the state of the art. Expert Systems with Applications, 121:49–65, 2019.

[12]. Sepp Hochreiter and Jurgen¨ Schmidhuber. Long short-term memory. Neural computation, 9:1735–80, 12 1997.

[13]. Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[14]. Pavan Kartheek Rachabathuni. A survey on abstractive summarization techniques. In 2017 International Conference on Inventive Computing and Informatics (ICICI), pages 762–765, 2017.

[15]. Usama Khalid, Mirza Omer Beg, and Muhammad Umair Arshad. RUBERT: A bilingual roman urdu BERT using cross lingual transfer learning. CoRR, abs/2102.11278, 2021.

[16]. Mohammad Khorasani, Mohamed Abdou, and Javier Hernandez´ Fernandez´. Streamlit Use Cases, pages 309–361. Apress, Berkeley, CA, 2022.

[17]. Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdel-rahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. CoRR, abs/1910.13461, 2019.

[18]. Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. AI Open, 2022.

[19]. Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. arXiv preprint arXiv:1908.08345, 2019.

[20]. N. Moratanch and S. Chitrakala. A survey on extractive text summariza-tion. In 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP), pages 1–6, 2017.

[21]. Nikita Munot and Sharvari S Govilkar. Comparative study of text sum-marization methods. International Journal of Computer Applications, 102(12), 2014.

[22]. Ramesh Nallapati, Bing Xiang, and Bowen Zhou. Sequence-to-sequence rnns for text summarization. CoRR, abs/1602.06023, 2016.

[23]. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text-transformer. CoRR, abs/1910.10683, 2019.

[24]. Radim Rehu˚ˇrek and Petr Sojka. Software Framework for Topic Mod-elling with Large Corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pages 45–50, Valletta, Malta, May 2010. ELRA.

[25]. Leonard Richardson. Beautiful soup documentation. Dosegljivo: https://www. crummy.com/software/BeautifulSoup/bs4/doc/.[Dostopano: 7. 7. 2018], 2007.

[26]. Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term

memory (LSTM) network. CoRR, abs/1808.03314, 2018.

[27].Sam Shleifer and Alexander M. Rush. Pre-trained summarization distillation. CoRR, abs/2010.13002, 2020.

[28].Ralf C. Staudemeyer and Eric Rothstein Morris. Understanding LSTM - a tutorial into long short-term memory recurrent neural networks. CoRR, abs/1909.09586, 2019.

[29].Qing Sun, Stefan Lee, and Dhruv Batra. Bidirectional beam search: Forward-backward inference in neural sequence models for fill-in-the-blank image captioning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 6961–6969, 2017.

[30].Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[31].Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi´ Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. CoRR, abs/1910.03771, 2019.

[32].Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi´ Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. CoRR, abs/1910.03771, 2019.

[33].Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. CoRR, abs/2010.11934, 2020.

Cite this article as :

Ketan Shahapure, Samit Shivadekar, Shivam Vibhute, Milton Halem, "Automated Text Summarization as A Service", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 11 Issue 1, pp. 54-65, January-February 2024. Available at doi : https://doi.org/10.32628/IJSRSET12310669
Journal URL : https://ijsrset.com/IJSRSET12310669