Extractive text summarization using Bi-LSTM and Meta-Learning

ABSTRACT

Text summarization is the task of condensing the number of sentences and words in a document without altering its meaning. Extractive Summarization method focuses on selecting the most crucial sentences within a text. The main problem of existing supervised text summarization models is that they require large datasets for training. Such datasets are very scarce. This project aims to make a model that works with minimal dataset with the help of Meta-learning strategies. To equip models with knowledge in both general text summarization and topic-specific summarization and to enhance adaptability to varied subjects this project utilizes Bi-LSTM model. Finally, we compare the developed model against contemporary algorithms using ROUGE score metrics.

INTRODUCTION

Nowadays, for any particular news, many articles are present in the web. Referring to each document on a whole and understanding each of them takes a lot of time and effort. Here is where summary plays a key role. Headlines in news, Chapter summaries in Textbooks, Movie/TV Show descriptions etc., have the responsibility of conveying the entire content in limited words while not disturbing the meaning of the content.

But, at the same time summary for each document cannot be generated manually as it also involves many minds working together putting a lot of effort while producing these summaries. Again, the main problem with the above approach is that there is no limit for the articles produced in the web. Therefore at some time, people have to wait for the summaries to be generated manually or will have to go ahead and read the entire article.

This problem therefore requires the attention of an automated model which will be able to produce summaries on demand of the users irrespective of the topic, domain or article. Summaries produced by humans might not always be accurate or effective and may require multiple reviews before being finalized. Unlike manual summaries, a well trained model for this task if used effectively, will produce appropriate summaries for the required data.

An Automatic text summarizer has the task of reducing the given data into a summary containing the core ideas while emphasising on the main points in the data in a minimum space and keep low repetitions. In addition to generating a summary, employing self learning techniques into the model will increase its performance as the model will be able to learn from its mistakes and improve itself. Hence we put forth our efforts in developing an automatic text summarizer which uses deep learning to produce summaries for a given text while utilising meta-learning for self learning.

RELATED WORKS

There are many existing models for automatic text summarization as it is not a new problem itself. Various works have been done to address this problem. Every model either uses extractive text summarization or abstractive text summarization. Extractive methods select the most important sentences within a text and therefore the result summary is just a subset of the full text, while the abstractive methods select the important points and generate its own summary by combining all the selected sentences to form a meaningful summary.

A Combined Extractive With Abstractive Model for Summarization(Wenfeng Liu) proposed hybrid extractive-abstractive two-stage summary generation model combines the strengths of both extractive and generative methods by first selecting important sentences using attention mechanisms and then restructuring them through syntactic analysis and beam search algorithm to produce the final summary.

In all the existing approaches for text summarization, many new methods have been introduced to tackle this task. Some of the models like Seq2Seq(Yong Zhang), COSUM(Rasim M. Alguliyev), Multi-Encoder Transformer(Youhyun Shin),BERT(Sheher Bano), RoBERT(Yuting Guo) introduced methods like Transfer learning, Chunk-based Framework, Word2Vec, Combinatorial Optimization, Evolutionary Optimization, Reinforced Learning etc., to text summarization task. All these models perform well when tested.Even though they're working well, all these models have a single common issue. That is, they require a large dataset for training. In general, these datasets are very scarce and are hard to obtain. Even if one obtains such a dataset, the model for sure takes a lot of time to train on the data to reach the level of expected accuracy. In contrast, Summerunner and NeuSum(Myeongjun Jangi) implemented a Training free model which is also language independent. But this model has limited performance, scalability issue and is vulnerable to noisy data all of which might be occurred due to the lack of proper training. Neural Attention Model(Aniqa Dilawari) uses Linguistic Feature Space for abstractive summarization but fails to identify distant dependencies.

Based upon the survey on the related works we have found some of the common issues and identified the problems which we want to overcome through our implementation of the model. The issues this model addresses are: large datasets required for training, capturing long distant relationships, and also employ a self-learning method for the mode to improve itself.

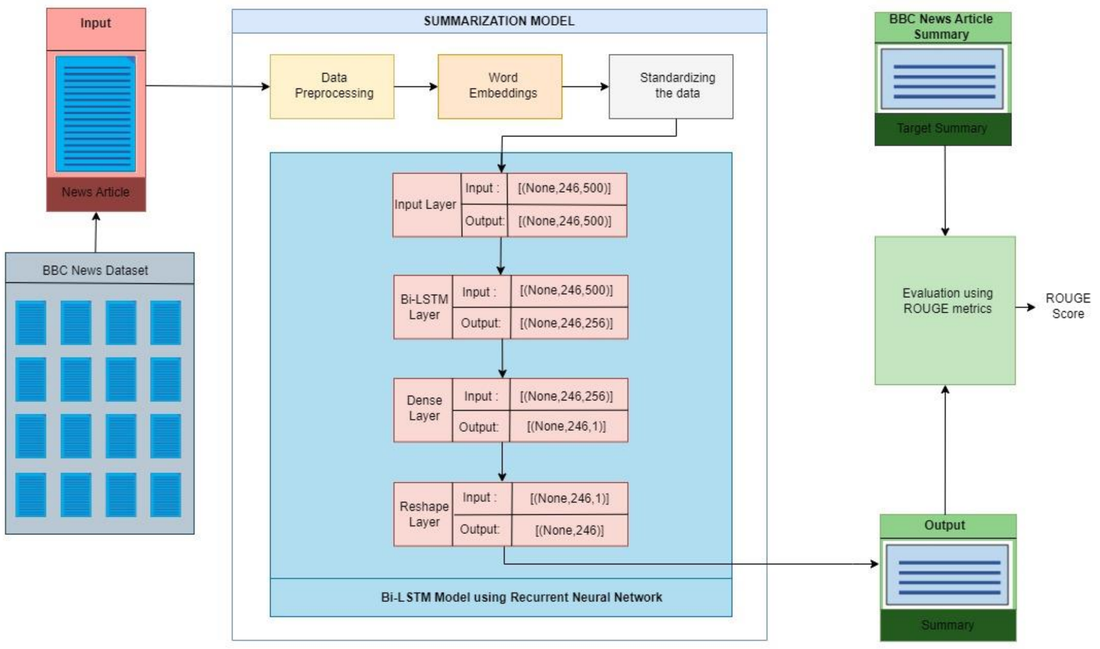
PROPOSED METHOD

Neural Networks deeply understand the meaning of the document and then summarize them based upon their training dataset. Their accuracy increases with increase in data size. But, RNN fails at the vanishing gradient problem. When this problem is encountered, the gradient of the activation function approached zero making the network hard to train. This means that even with a large change in the input data, there will be minimal to no change in the output data. This will not be ideal and the model should overcome this.

LSTM models can overcome the problem of vanishing gradients. Keeping the above aspects in mind, we develop a Bi-LSTM model which utilises RNN. The Bi-LSTM model unlike traditional RNNs, has the capability to remember long term dependencies between words. It has memory cells which help the model to remember this relationships. Also, it processes the input in both forward and backward directions extracting and understanding the deeper context of the document. The Bi-LSTM model combined with activation functions like ReLU etc., solve the vanishing gradient problem.

Again, one of the other main problem is the datasets required for training.These networks require large amount of datasets to be trained which are difficult to find and even if are to be found, the model takes a lot of time for training on the data. The Bi-LSTM model also doesn't demand a large dataset for training. It produces effective results even with a limited training data.

Additionally self learning techniques such as Meta-Learning will help the model to learn by itself and the model gets better each time it produces a summary. For this purpose, MAML – Model Agnostic Meta-Learning algorithm is employed. This approach also helps the model to generate topic based summaries on the given text.



Architecture Design

Meta-Learning strategy:

Algorithm Model-Agnostic Meta-Learning

Require: N support datasets: $T_S = \{T_{S1}, T_{S2}, ..., T_{Sn}\}$.

Require: The query dataset: T_Q

Require: The learning rate hyperparameters: α, β

```
1: randomly initialize $\theta$
2: for $e$ iterations do
3:     Sample batches of the dataset, each with a size of $m$.
4:     for all $T_{S_i}$ do
5:         Evaluate $\nabla_\theta L_{T_{S_i}}(f_\theta)$ with respect to $m$ examples.
6:         Compute adapted parameters with gradient descent:
           $\theta_i' = \theta - \alpha \nabla_\theta L_{T_{S_i}}(f_\theta)$
7:     end for
8:     Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{i=1}^{N} L_{T_Q}(f_{\theta_i'})$
9: end for
```

Here,

- $\boldsymbol{\beta}$ = learning rate (Adam Optimization), $\boldsymbol{\alpha}$ = Initial learning rate, $\boldsymbol{\theta}$ = initialized model parameters, $\boldsymbol{\theta'}$ = initialized model parameters, **e** = no' of epochs, **N** = no' of support tasks,
- $\boldsymbol{L_{T_{S_i}}(f_\theta)}$ is the binary cross-entropy loss function for the model $\boldsymbol{f_\theta}$ on the support dataset $\boldsymbol{T_{S_i}}$.
- $\boldsymbol{L_{sum} = \sum_{i=1}^{N} L_{T_Q}(f_{\theta_i'})}$ is calculated by adding the sum of all the losses using binary cross entropy obtained on the parameters $\boldsymbol{\theta_i'}$.
- $\boldsymbol{L_{sum}}$ is used to perform meta update on the model using $\boldsymbol{\theta^* = \theta\ \beta \nabla_\theta L_{sum}}$.

Time complexity of MAML:-

- **O(e x b_Q x N x b_S)**

  Where $\mathbf{b_S}$ = no' of training batches of support dataset and $\mathbf{b_Q}$ = no' of training batches of query dataset.

Reasons for choosing MAML:-

- **Model Agnostic:** applied to almost all models
- **Gradient-Based Adaptation:** helps the model to rapidly adapt to new tasks.

By implementing this procedure, the model will now be able to learn by itself and adapts to new tasks such as topic based summarization, query answering etc.,

DATA COLLECTION

Our team has decided to use BBC News Summary dataset which is available at http://mlg.ucd.ie/datasets/bbc.html. The dataset consists of 2225 news articles under 5 categories such as business, tech, entertainment, politics and sport along with their corresponding summaries.

Additionally there is a pre-processed version of the dataset that contains files having metadata regarding the data set such as

1. A .mtx file that stores the original term frequencies.
2. A .terms file that contains the unique content bearing words in the dataset with each row in .terms file corresponding to each row in .mtx file.

We utilised the **.terms** file to develop a **term dictionary** for our embedding task.

IMPLEMENTATION

Data Preprocessing:

From the obtained raw text data, we implemented several preprocessing steps to convert the data into the required form to be able to train the model. The follow steps were followed sequentially on the News Article and the summary alike.

Lowercasing the data: The entire text is converted into lowercase so that it won't be necessary to treat capitalized words and normal words as separate words. It avoids complexities while processing the input.

Combining the document into a single paragraph: The document is made rid of new line characters and stripped of unwanted spacings and is combined into a single paragraph.

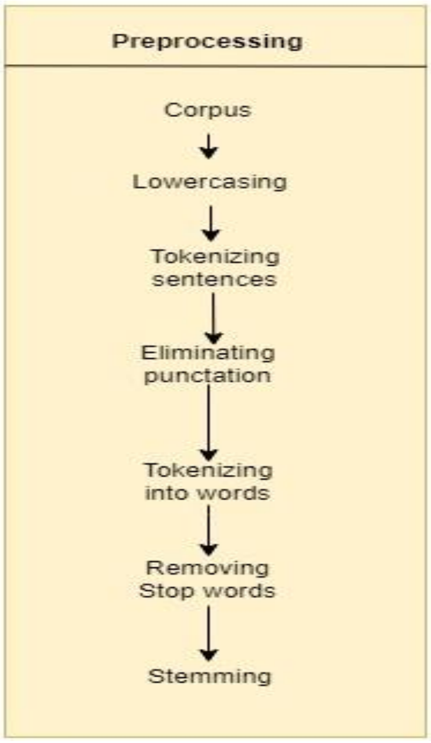
Tokenizing the Text into sentences: We used the sent_tokenize method to split the text into sentences.

Eliminating punctuation: Upon obtaining the sentences, in each sentence, we have removed punctuation such as "!" , " ." , "," etc.,

Tokenizing the data: We tokenized the sentences into words using word_tokenize method.

Removing Stop words: We then removed the stop words from the tokenized data so that only the words related to content are present. The stopwords are chosen by the team itself and consists of 300 words which were not considered from the dataset when the .terms file was built. The list is available in the preprocessed version of the dataset. So, we chose to use that list of stopwords in our implementation.

Stemming the words: We have used PorterStemmer to stem the tokenized words so that we will be able to match them with the data dictionary as it is the mentioned process that was followed on every word which was stemmed and added to the .terms file.



Word Embeddings:

As mentioned earlier, We have used a data dictionary to embed our words as integers. The pre-processed version of our chosen dataset included a .terms file which we have utilised as a dictionary. It consisted of 9635 unique entries after filtering out stop words(300) and filtering out words with low frequency count(<3). So, we considered each row for each word having and taken the index value (starting from 1) as the corresponding integer value for the word. Since this document has stemmed words, we have also chosen to stem words using the PorterStemmer so that the words in the text will match with the dictionary.

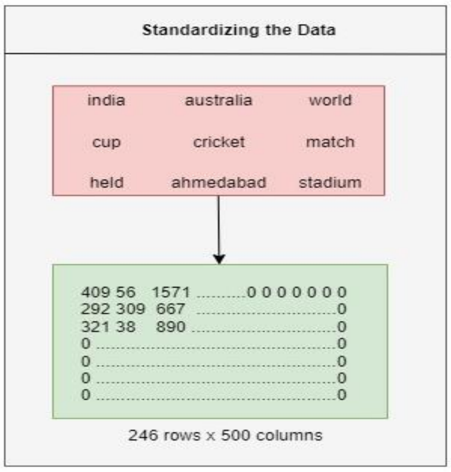
Data Preparation:

Based upon the facts that our team has gathered regarding the dataset we are using, the BBC News Articles dataset consists 2225 articles out of which the highest number of sentences in an article is 246 and the highest number of words in a sentence is 487. So, we wanted to standardize the input data into a 246 X 500 input matrix which will both be able to accommodate the article with 246 sentences and also the sentence with highest number of words or both even at the same time.

For this purpose, we initially declared a 246 X 500 matrix with every entry being zero using numpy's np.zeroes() method. Then, we mapped each of the obtained entry for the text i.e, the embedded integer into their respective positions in the standard matrix. After this, we will have sparse matrix representing the given text as integers in a 246 X 500 format.

Note that every article follows the same pattern and the input to the model will always be in a standard format.

But for the summary file, the expected summary from the model is initially resulted as a 246 element array each element being 1 if that particular sentence in the article belongs in the summary or 0 if not. So, the expected summary was built from the news article file by initially checking whether a sentence belongs to the target summary or not and the putting a 0 or 1 in that place. In this way for every article, 246 elements are created as an array.



For the training, testing and validation purpose, we have split the dataset into 3 categories as 80% for training purpose and 10% each for validation and testing purpose. Separate CSV files for the three categories were created using a preparation function developed by the team. This involved in taking the input article and separating the headline from it and taking the corresponding summary from the dataset and putting all the three values in a data frame. This process was followed for all the articles present in the dataset. Finally, CSV files were produced which contained the data as headline, article, summary format. We generally did not bother about the headline column as we wanted to generate summary on the article.

The training and validation files were used to train and validate the model while at last we test the model using the testing file. These files consisted 1776, 227 and 221 articles respectively.

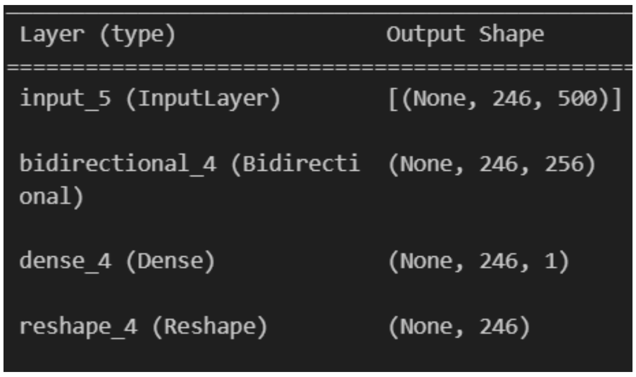
Dataset preparation for MAML-Training

Aside from the non-MAML version of the implementation where the model is fed with the entire dataset at once and tested on the entire dataset, MAML requires task based approach which introduces variety to the model training. This involves in separating the dataset into separate tasks each representing a unique topic or task. We followed the same approach and utilised the already segregated BBC dataset as training dataset for MAML model training.

For MAML we use two types of datasets. One dataset is support dataset and the other being query dataset for each of the tasks for which the dataset is segregated. The support dataset is used for model training and query dataset is used for model testing. Based upon the facts from the dataset, we chose the support dataset size as 300 texts while query dataset size as 80 texts as minimum articles in the dataset for a given topic is 386.

Building The Model:

The model we want to construct consists of four layers namely the Input Layer, Bi-directional LSTM Layer, Dense Layer and Reshape Layer.



```
Layer (type)                 Output Shape
==============================================
input_5 (InputLayer)         [(None, 246, 500)]

bidirectional_4 (Bidirecti   (None, 246, 256)
onal)

dense_4 (Dense)              (None, 246, 1)

reshape_4 (Reshape)          (None, 246)
```

The Input layer takes the pre-processed texts as input. This is a matrix of integers in the chosen 246 X 500 format.

The BiLSTM layer takes the input from the input layer(246X500) and process the data in both forward and backward directions. Here is where the context of the article is extracted. The BiLSTM model excels in finding the dependencies of the data as it processes the data in both the directions and it can analyze the in depth meaning of the data. The units chosen are 128 for each LSTM and since this version is Bidirectional, it becomes 256 units overall for the Bidirectional layer. These units help in performing the computations on the previous layer output i.e., this layer's input. This layer produces output in the shape (246,256)

The Dense layer takes output of Bidirectional layer as input and decides whether the provided sentence needs to be in the summary or not. This is done by predicting the probability of each sentence whether it needs to be in the summary or not. The output is of the form (246,1)

The Reshape layer is used to transform the output of Dense layer to a one-dimensional vector and is used to evaluate the model, compute loss and for optimization purposes. The output of this layer is of the form (246,)

Finally the result from the Reshape layer is taken and the summary is generated by selected by taking the sentences that are predicted to be in the summary.

The model is optimized using Adam optimization and we set the learning rate as 0.005 as it produced best results.

Implementing Model-Agnostic Meta-Learning

MAML training involves in training an original model replica using support dataset, measuring the losses obtained in the training process and performing inner updates on the replica model by applying the gradients calculated for the losses to optimize the model for further training. After the training for a task is completed, the losses obtained on the replica model for the query dataset are measured and the corresponding gradients are calculated after which we perform a meta-update on the original replicated model to optimize the model to provide summaries nearer to the target summaries.

So, based upon this approach we first clone the model which is initiated with some random parameters. We call this model as *inner_model.* We initially perform training on the model for a support dataset and then compute losses during the training and then calculate the gradients for the training variables according to the obtained losses after which we perform inner update on the inner_model by optimizing the inner_model by applying the gradients to the inner_model's optimizer. Once the support training is completed for the inner_model for the specifed epochs(inner_epochs), testing is done for the inner_model using the query dataset and then loss for the query dataset is calculated and then we calculate the gradients for the inner_model's variables on query dataset loss and apply this gradients to the original model's optimizer.

The inner_model's learning rate was set at 0.001 and the original model's learning rate was also set as 0.001 as they produced the best results upon testing.

Generating Summary from model's output

The output of the model is the output of the reshape layer which is a 246 element array which contains the probability of each of the 246 sentences being a part of the summary or not. Generally these values vary in the range from 0 to 1, but we chose a threshold value by which we decide whether a sentence needs to be in the summary or not. If initially, the predicted probability of the sentence is greater than this threshold value of 0.40, the sentence belongs to the summary, else it doesn't. Now, since the probabilities for every sentence in every article vary according to the dataset, we update the value of this threshold by Exponential Weighted Moving Average procedure. In this procedure, we update the threshold value by considering a percent of weight from both previous value i.e, the threshold value and the remaining percent of weight from the probability average of each article that the model has worked on to predict summary. But keeping a case in mind where there might be lesser or even no such sentences, we kept the minimum number of sentences on a summary as 5. So, under a situation where the summary sentences is less than 5, sentences with higher probabilities when compared to other sentences are joined in the summary until the number of sentences in the summary become equal to 5. The point to be noted is with EWMA consideration for generating summaries, the model has produced better summaries because of stricter threshold value. We have considered 999% of threshold value combined with 1% of probability average of each article by updating the threshold value for each article.

We take the values greater than the threshold value from this output and keep it in a list and finally sort it in descending order. The next step involves in extracting the index of the chosen sentences from this predicted output list and then match the index to the corresponding input text(which we converted into list of sentences) element and then including it in the summary. All such sentences are joined together to form a summary.

Loss computation: Loss is computed as mean loss for a support dataset using binary cross entropy function upon the predicted data by the inner_model.

Now, we perform few-shot learning on this inner_model. We can choose to perform normal training on this model but it can consume more time for the entire process to complete. In contrast, through gradient updation, few-shot learning provides the same results when compared to normal training but in much lesser time.

TRAINING, VALIDATION& TESTING

The model has been trained under different conditions such as 20, 30, 50 and 100 epochs under various batch sizes such as 9, 20, 30 and 50. The learning rate also varies as 0.001 and 0.005 for comparison purposes.

Initially we chose the metrics as accuracy and tested the model on the above mentioned conditions. The model performed well at 50 epochs and batch size as 9. Based on our observations, even though 100 epochs produced a very little improvement over 50 epochs, we believe that it might overfit the model. Hence, we did not consider 100 epochs. Training on 20 and 30 epochs did not provide better results when compared to 50 epochs. Further we varied the batch sizes which made considerable changes in the model accuracy. The model produced its best result at batch size as 9 when compared to other tests. The learning rate made no changes as it only helped in speeding up the learning process which at the end made no change to model's performance. At last, we tested model's performance on the training data and obtained similar result as training.

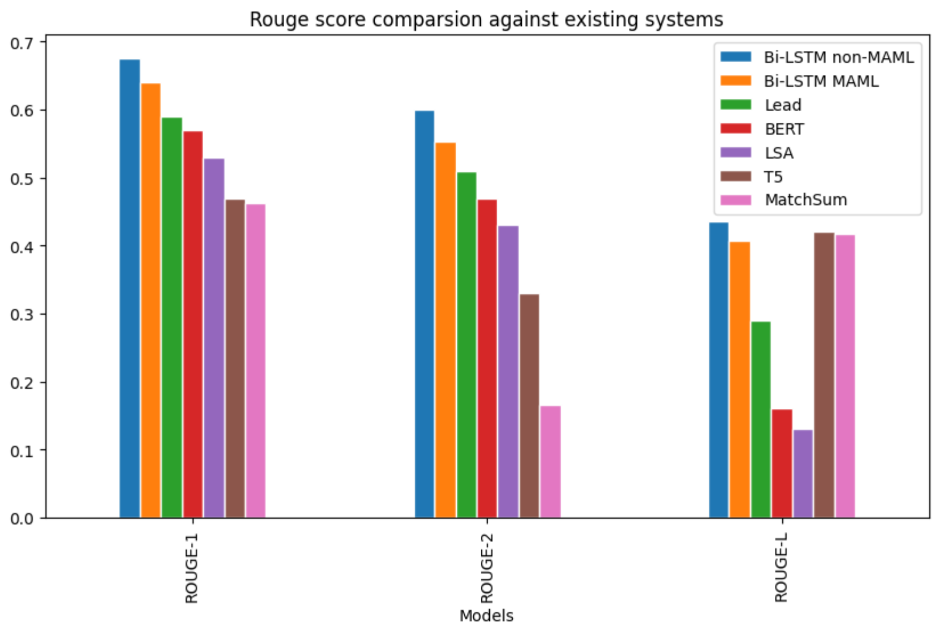
Few-Shot Training:

For few shot training, we train the model on N-Shots with N varying as 5, 10, 15, 20 on 5, 10, 20, 30 epochs. While the support dataset still consists of 300 texts, only N texts are selected at random for training, but testing is done on the entire query dataset. The inner_model is also trained on inner_epochs which can be varied as required, but more inner_epochs demand more time for training. We chose the inner_epochs as 5, 10, 15 for our inner_model.

Training and testing has been done on various conditions to get an insight on the model's performance under various conditions to leverage model's best performance. From our observations, the model with MAML worked its best at 30 epochs, 10 inner_epochs and 5-shot data.

RESULT & EVALUATION

As mentioned earlier, we aim to compare the model's performance using ROUGE-metrics. For this, we took the target summaries in a list as references and predicted summaries of the model as predictions and computed the ROUGE scores of the model. Below are the obtained scores:



Non-MAML Scores:

ROUGE-1:	**ROUGE-2:**	**ROUGE-L:**	**ROUGE-Lsum:**
0.676207931	0.599503236	0.436380759	0.436451758

MAML Scores:

ROUGE-1:	**ROUGE-2:**	**ROUGE-L:**	**ROUGE-Lsum:**
0.636085006	0.534872752	0.402168444	0.402031715

Model's performance compared against similar models for Extractive Text Summarization

MODEL	ROUGE-1	ROUGE-2	ROUGE-L
Non-MAML	**0.67**	**0.59**	**0.43**
MAML	**0.636**	**0.53**	**0.40**
T5(Anushka Gupta, 2021)	0.47	0.33	0.42
MatchSum(Ming Zhong, 2020)	0.462	0.165	0.418
LSA	0.53	0.43	0.13
Lead(Begum Mutlu, 2020)	0.59	0.51	0.29
BERT (Devlin et al., 2018)	0.57	0.47	0.16
Fusion Sum(Liqiang Xiao, 2022) (CNN)	0.425	0.191	0.387
DANCERLSTM(Alexios Gidiotis, 2020) (CNN)	0.440	0.176	0.402

For this purpose, we appended all the generated summaries into a list and reference summaries into another list. Finally, we computed the model using rouge metrics on the predicted summaries against reference summaries.

From the above comparison table, we have run all the models on BBC summary datasets to perform summarization and the remaining models have used CNN news dataset for the same purpose. Our model has outperformed all the models in ROUGE-1, ROUGE-2 scores category for both non-MAML and MAML versions while in ROUGE-L, the non-MAML version produced the best results while the MAML version was outperformed by T5 and MatchSum.

Additionally, when compared against models utilising LSTM on extractive summaraization, our model outperformed DancerLSTM which was developed to perform extractive text summarization on the CNN dataset and the similar goes to FusionSum.

CONCLUSION

Through this experiment we performed extractive text summarization using Neural Network with only a limited dataset. For this purpose we utilised Bidirectional-LSTM and its capability to remember long term dependencies. We performed the experiments on BBC news articles and summaries dataset and computed the result using ROUGE metrics. When the obtained score is evaluated against existing models, our model has produced significantly better results with only limited dataset. The results thus prove that when we have a limited dataset, LSTMs perform well and produce better results when compared other models.

Based on our observations, we believe that due to Network's ability to understand the deeper context of the sentences combined with LSTMs memory cells helping a network to identify and remember long term dependencies are the reason for the model's resultant performance. We strongly believe that if the model is further trained on a new unseen data, its performance will improve significantly.

Additionally, we implemented Meta-Learning on this model to improve the model's performance further more as it helps model to learn and adapt even more quicker as we aimed to work on a limited dataset. When performed few-shot training, the model has produced effective results in comparatively lesser time. This proves that by utilising MAML, models can be made to be rapidly adaptable for new tasks. We believe further modifications and additional updations to the algorithm might significantly improve the model's performance which we aim to do.

REFERENCES

Greene, D., & Cunningham, P. (2006a). BBC datasets. http://mlg.ucd.ie/datasets/bbc.html.

[1] W. Liu, Y. Gao, J. Li and Y. Yang, "A Combined Extractive With Abstractive Model for Summarization," in IEEE Access, vol. 9, pp. 43970-43980, 2021, doi: 10.1109/ACCESS.2021.3066484.

[2] M. Jang and P. Kang, "Learning-Free Unsupervised Extractive Summarization Model," in IEEE Access, vol. 9, pp. 14358-14368, 2021, doi: 10.1109/ACCESS.2021.3051237.

[3] Y. Zhang and W. Xiao, "Keyphrase Generation Based on Deep Seq2seq Model," in IEEE Access, vol. 6, pp. 46047-46057, 2018, doi: 10.1109/ACCESS.2018.2865589.

[4] Alguliyev, R., Alıguliyev, R. M., Isazade, N. R., Abdi, A., & Idris, N. (2018). Cosum: text summarization based on clustering and optimization. Expert Systems, 36(1). https://doi.org/10.1111/exsy.12340

[5] A. Dilawari, M. U. G. Khan, S. Saleem, Zahoor-Ur-Rehman and F. S. Shaikh, "Neural Attention Model for Abstractive Text Summarization Using Linguistic Feature Space," in IEEE Access, vol. 11, pp. 23557-23564, 2023, doi: 10.1109/ACCESS.2023.3249783.

[6] Y. Shin, "Multi-Encoder Transformer for Korean Abstractive Text Summarization," in IEEE Access, vol. 11, pp. 48768-48782, 2023, doi: 10.1109/ACCESS.2023.3277754.

[6] Nabil Alami, Mohammed Meknassi, Noureddine En-nahnahi, Enhancing unsupervised neural networks based text summarization with word embedding and ensemble learning, Expert Systems with Applications, Volume 123, 2019, Pages 195-211, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2019.01.037.

[7] S. Bano and S. Khalid, "BERT-based Extractive Text Summarization of Scholarly Articles: A Novel Architecture," 2022 International Conference on Artificial Intelligence of Things (ICAIoT), Istanbul, Turkey, 2022, pp. 1-5, doi: 10.1109/ICAIoT57170.2022.10121826.

[8] Guo, Y.; Ge, Y.; Yang, Y.-C.; Al-Garadi, M.A.; Sarker, A. Comparison of Pretraining Models and Strategies for Health-Related Social Media Text Classification. Healthcare 2022, 10, 1478. https://doi.org/10.3390/healthcare10081478

[9] Gidiotis, Alex & Tsoumakas, Grigorios. (2020). A Divide-and-Conquer Approach to the Summarization of Long Documents.

[10] Liqiang Xiao, Hao He, Yaohui Jin, FusionSum: Abstractive summarization with sentence fusion and cooperative reinforcement learning, Knowledge-Based Systems, Volume 243, 2022, 108483, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2022.108483.

[11] [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](https://aclanthology.org/N19-1423) (Devlin et al., NAACL 2019)

[12] Mutlu, Begum & Sezer, Ebru & Akcayol, M.. (2019). Multi-document extractive text summarization: A comparative assessment on features. Knowledge-Based Systems. 183. 10.1016/j.knosys.2019.07.019.

[13] [Extractive Summarization as Text Matching](https://aclanthology.org/2020.acl-main.552) (Zhong et al., ACL 2020)

[14] Gupta, Anushka & Chugh, Diksha & Anjum, Anjum & Katarya, Rahul. (2021). Automated News Summarization Using Transformers.