Summarization of Long Input Texts Using Multi-Layer Neural Network

Niladri Chatterjee, Aadyant Khatri and **Raksha Agarwal**
Indian Institute of Technology Delhi
Hauz Khas, Delhi-110016, India
{niladri@maths.iitd.ac.in, tt1191@iitd.ac.in, maz178296@maths.iitd.ac.in}

Abstract

The present paper describes the architecture of a novel Multi-Layer Long Text Summarizer (MLLTS) system proposed for the task of creative writing summarization. Typically, such writings are very long, often spanning over 100 pages. Summarizers available online are either not equipped enough to handle long texts, or even if they are able to generate the summary, the quality is poor. The proposed MLLTS system handles the difficulty by splitting the text into several parts. Each part is then subjected to different existing summarizers. A multi-layer network is constructed by establishing linkages between the different parts. During training phases, several hyper-parameters are fine-tuned. The system achieved very good ROUGE scores on the test data supplied for the contest.

1 Introduction

Summarization of long texts is a challenging problem for different widely available summarizers. While the Deep Learning (DL) based summarizer BART (Lewis et al., 2019) is severely restricted by the size of the input document, the quality of other traditional summarizers, namely LexRank (Erkan and Radev, 2004), TextRank (Mihalcea and Tarau, 2004) are found to be poor in terms of different ROUGE scores. The present paper proposes a novel architecture, Muti-Layer Long Text Summarizer (MLLTS), to overcome the above challenge. Figure 1 provides a schematic diagram of the proposed architecture.

The novelty of the MLLTS architecture is that it uses multiple online summarizers for carrying out the summarization task in the following way. First, the long input text is partitioned into several parts. These parts are assigned to different layers of the multi-layer architecture. If the document is partitioned into p parts and s is the number of summarizers used, then the total number of layers is $p \times s$. Different parameters are trained to fine-tune the intra-layer and inter-layer connections to optimize the overall output. Finally, VoteSumm method proposed by Agarwal and Chatterjee (2022) is used for generation of the summary, by combining different part summaries in an optimized manner.

The rest of the paper is organized as follows. Section 2 provides a brief review of some past works on network-based summarization. Section 3 presents a detailed description of the proposed architecture. Sections 4 and 5 describe the experiments conducted and the results obtained, respectively.

2 Related Past Works

Graph-based sentence ranking is a popular text summarization technique. In this approach, the input text is represented using a graph/network of sentence nodes. Edges between the nodes are created to represent the relationship between them. The nodes are ranked using different methods to determine their overall importance with respect to the given input text. Finally, the summary is generated by selecting the sentences which receive high ranks.

Mihalcea and Tarau (2004) used lexical similarity between sentences for edge creation in the graph and used PageRank (Brin and Page, 1998) to rank the nodes. Erkan and Radev (2011) used cosine similarity between bag-of-words representation of sentences for adding edges in the graph.

Tohalino and Amancio (2017) performed a summarization of multiple similar documents using multilayer networks. The layers of the network are used to represent each individual input document. TF-IDF based cosine similarity is used for connecting sentence nodes. Node ranking is performed via nine different network measurements, such as degree, strength, PageRank, accessibility, and symmetry, among others, for generation of summaries.

Alzuhair and Al-Dhelaan (2019) used a combination of different edge weighting schemes, namely

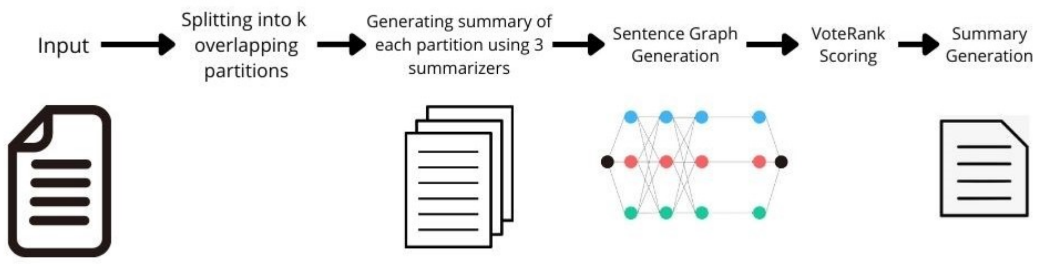
Input
Splitting into k overlapping partitions
Generating summary of each partition using 3 summarizers
Sentence Graph Generation
VoteRank Scoring
Summary Generation

Figure 1

Jaccard similarity, TF-TDF similarity, Topic signature similarity and Identity similarity along with PageRank and HITS (Kleinberg, 1999) node ranking methods for summarization.

3 Proposed Approach: Multi-Layer Long Text Summarizer

The proposed MLLTS approach works in the following way. First, the long input text is partitioned into p smaller parts. These parts are not discrete, rather there is a 25% overlap between two successive parts. Then, each part is summarized using different summarizers. We have used three summarizers for the present work: TextRank, LexRank and Distil-BART.

The $3 \times p$ short texts thus formed, from the three summaries for each of the p parts for a long input text, constitute the different layers of a network graph. The sentences from these $3 \times p$ short texts form the nodes of the network graph and weighted edges are added between the nodes based on Jaccard similarity of the corresponding sentence vectors. Three parameters are fine-tuned to optimize the outputs. These are:

- K: used as a threshold to select a subset of edges of the network.
- α: that optimizes the strength of connections between sentences belonging to different parts.
- β: that finetunes the connection between sentences coming through the same summarizer.

For α and β, a value > 1 implies they strengthen the connections. Similarly, values < 1 weaken the strength. Once the network is prepared, VoteSumm technique is used to rank the nodes. The highest-ranking nodes are then used to generate the final summary.

However, one major challenge still faced is that the size of some partitions is still too long for Distil-BART to generate a summary. Hence, we introduce the following novelty while partitioning the long input text. In such cases, the long input text is not partitioned into the p fixed number of parts. Rather, the partitioning is such that all the parts have a fixed number of words. The number of words is chosen such that DistilBART can generate a summary on it.

In our experiments, we discovered that the summaries generated by a fixed number of partitions exhibit better ROUGE scores. Hence, we kept this method to be the default option for partitioning. Partitioning on a fixed number of words is used only in situations discussed above.

4 Experimental Details

4.1 Dataset

The training data provided for the BookSum component of the Automatic Summarization for Creative Writing contest, was split into two parts, train-split, and validation-split. The train-split part had a total of 6759 samples, whereas the validation-split had 984 samples. The samples in the training set spanned 148 different books and 4931 chapters, with an average of 5424.32 words and 169.23 sentences. The corresponding target gold summaries had 362.26 words and 23.32 sentences on an average. There were 17 unique books and 636 chapters in the validation-split. These samples had 5097.17 words and 214.83 sentences on an average. Their gold summaries had 157.58 words and 10.35 sentences. The various statistics can be seen in Table 1.

4.2 Experiment – 1

As the first experiment, several summarizers provided in the SUMY package [https://

	Train-Split Inputs	**Train-Split Target Summaries**	**Validation - Split Inputs**	**Validation - Split Target Summaries**
Avg. no. of words	5424.32	362.26	5097.17	157.58
Max. no. of words	17928	2442	11010	741
Min. no. of words	754	41	899	23
Median no. of words	4523	307	4873	126
Avg. no. of sentences	169.23	23.32	214.83	10.35
Max. no. of sentences	591	126	562	48
Min. no. of sentences	10	2	9	2
Median no. of sentences	148.5	19	187.5	10

Table 1

github.com/miso-belica/sumy], a module for automatic summarization of text documents, were used separately on the input texts to generate the summaries. More specifically, we used the following summarizers:

- TextRank Summarizer
- Sum Basic Summarizer
- LSA Summarizer
- LexRank Summarizer
- Random Summarizer

Based on the ROUGE scores obtained, TextRank and LexRank summarizers were selected and used in the subsequent experiments. Table 2 provides the results obtained through Experiment-1. The ROUGE scores tabulated are the average scores obtained by different parameter sets over several input samples.

Summarizer	**ROUGE 1**	**ROUGE 2**	**ROUGE SU4**
TextRank Summarizer	0.142	0.018	0.033
Sum Basic Summarizer	0.140	0.016	0.033
LSA Summarizer	0.132	0.015	0.029
LexRank Summarizer	0.125	0.014	0.027
Random Summarizer	0.111	0.011	0.025

Table 2

4.3 Experiment – 2

In the second experiment the input was split into p parts. Then the above summarizers were used on each of the parts. For future reference these summaries will be called sub-summaries. These sub-summaries were then combined to obtain the summary for the whole text. In particular, we have worked with the values of $p \in \{3, 5, 8, 10\}$

The scores obtained by the whole text summaries thus generated from each of the summarizers were compared. It was hypothesized that using this method, important information from various parts of the input will be considered while generating the final summary. Results obtained through Experiment - 2 are given in Table 3.

Summarizer	**ROUGE 1**	**ROUGE 2**	**ROUGE SU4**
LexRank Summarizer	0.126	0.008	0.027
TextRank Summarizer	0.121	0.008	0.025

Table 3

4.4 Experiment – 3

As a variation to Experiment - 2, instead of directly appending the sub-summaries, the sub-summaries were fed into the VoteSumm technique to produce the final summary for the input. Apart from the number of partitions p, VoteSumm has two more hyper-parameters, K and α (as defined in Section 3).

We experimented with different sets of values for these parameters as given below:

$\boldsymbol{K} \in \{0.1, 0.25, 0.5, 0.8\}$
$\boldsymbol{\alpha} \in \{0.25, 0.3, 0.75, 1, 1.25, 1.5, 1.75, 2.0\}$

Along with TextRank and LexRank summarizers, state-of-the-art transformer-based summarizer (DistilBART) was also used. Table 4 contains the best five performing parameter sets in Experiment - 3.

α	*p*	*K*	ROUGE 1	ROUGE 2	ROUGE SU4
1.0	3	0.1	0.230	0.028	0.053
0.5	3	0.1	0.223	0.020	0.049
0.1	3	0.1	0.216	0.023	0.047
1.5	3	0.1	0.208	0.020	0.046
1.5	3	0.8	0.193	0.023	0.044

Table 4

4.5 Experiment – 4

In this experiment, different summarizers were no longer treated separately. Once the input was split into *p* parts, and the summaries for each of the parts were generated using the three summarizers (namely, LexRank, TextRank and DistilBART), all the 3 × *p* sub-summaries were treated as different layers of VoteSumm to generate the combined summary. As before, *p*, *K* and α were varied, and the scores produced were compared. Along with these, a new hyper-parameter β was introduced (as defined in Section 3).

With the introduction of parameter β in conjunction with α, we experimented with two different combinations of them, namely ($\alpha + \beta$, $\alpha \times \beta$) to use them in VoteSumm. The following values of β were experimented with:

$$\boldsymbol{\beta} \in \{0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2.0\}$$

Our experiment here was to measure the efficacy of these two schemes in generating the whole-text summaries. Table 5 shows the five best performing parameter sets based on Experiment - 4.

α	β	*p*	*K*	Formula	ROUGE 1	ROUGE 2	ROUGE SU4
0.25	1.5	8	0.1	multiply	0.159	0.013	0.034
0.25	1.75	8	0.1	multiply	0.158	0.014	0.034
0.25	2.0	8	0.1	multiply	0.157	0.013	0.033
0.25	1.25	8	0.1	multiply	0.151	0.012	0.032
0.25	1.0	8	0.1	multiply	0.149	0.012	0.031

Table 5

4.6 Experiment – 5

In all the above experiments, the input was split into several discrete partitions. In this experiment, the input was split into *p* overlapping partitions. The rest was kept the same as in Experiment - 4. As a result of overlapping portions, a new hyper-parameter, 'partition-percent', came into consideration. The role of this hyper-parameter was to decide the percentage of overlap to be present in the partitions. In particular, we tried with values of 15% and 25%. The results of Experiment - 5 are given in Table 6.

α	β	*p*	*K*	Partition Percent	Formula	ROUGE 1	ROUGE 2	ROUGE SU4
0.3	2.5	8	0.1	25%	multiply	0.108	0.007	0.023
0.3	1.5	8	0.1	25%	multiply	0.108	0.007	0.023
0.3	2.0	8	0.1	25%	multiply	0.106	0.006	0.023
0.3	2.0	8	0.1	25%	add	0.105	0.006	0.023
0.3	2.5	8	0.1	25%	add	0.104	0.006	0.023

Table 6

4.7 Experiment – 6

While conducting Experiment – 5, we noticed that a significant number of sub-summaries could not be computed by DistilBART because of its limited input size. To overcome this, we introduced the following novelty in the proposed MLLTS architecture.

Instead of having a fixed number of splits for each of the input samples, the size of the split was fixed. This implied that the number of partitions may be different for different input texts as their sizes vary significantly. Hence in this experiment, the sizes of the partitions remained the same for all the input samples. Overlapping was also present in these partitions. Further, the three summarizers were used to generate summaries of *p* (not decided beforehand) parts of an input and then these 3 × *p* sub-summaries were fed to VoteSumm. As before we experimented with different choices of the hyper - parameter values. The results of Experiment - 6 are in Table 7. Section 5 analyzes the results of different sets of experiments.

α	β	K	Partition Percent	Formula	ROUGE 1	ROUGE 2	ROUGE SU4
0.3	2.5	0.1	25%	multiply	0.116	0.008	0.026
0.3	1.5	0.1	25%	multiply	0.115	0.008	0.025
0.3	2.0	0.1	25%	multiply	0.115	0.008	0.025
0.3	2.5	0.1	25%	add	0.112	0.007	0.024
1.2	2.5	0.1	25%	multiply	0.112	0.006	0.024

Table 7

5 Results and Discussions

The results in the form of ROUGE scores obtained from Experiment - 1 and Experiment - 2 were not up to the mark. This was not surprising, and was rather intuitive since summarizers were directly applied to the input. However, there was a significant increase in the ROUGE scores from Experiment - 3 onwards. As given in Table - 4 the best five performing sets of parameters α, K and p are shown.

From the results of Experiment - 4, it became evident that having $p = 8$ and $K = 0.1$ yielded better results. Multiplying α and β to combine them and using higher values of β along with it also led to higher scores.

Once overlap was introduced between different partitions, combining α and β through addition also led to good scores. However, it was noted that having just 15% overlap did not improve the results but an overlap of 25% improved the results significantly. The best values for K and p again came out to be 0.1 and 8, respectively.

In Experiment – 6, where, p, the number of splits, was not pre-decided, the best results were obtained when α and β were multiplied to combine them, high values of β were used and K was set to be 0.1.

6 Conclusion

This paper proposes Multi-Layer Long Text Summarizer (MLLTS) as a possible solution of one of the modern day information processing problems, namely, summarization of long texts. Although a large number of summarizers have been developed and made available over the last decade, their performance on long text is highly questionable. The novel MLLTS system proposed in this work is our contribution towards this need. This is specially designed for the BookSum component of the Automatic Summarization for Creative Writing contest, COLING 2022. The input texts for this system were chapters from famous English books. The corresponding target summaries were expected to be of the order of 5% of the input text size.

A series of experiments were carried out to fine-tune several hyperparameters that are associated with the architecture. The successive design decisions and experiments are so planned that steady improvements in performance, with respect to ROUGE scores, can be observed. The best model, as per performance on the validation set, obtained values of 0.159 and 0.014 for ROUGE-1 and ROUGE - 2, respectively. However, on the test data, it achieved much better scores. For the test data, the ROUGE-1 and ROUGE-2 scores obtained were 0.2643 and 0.0471, respectively. Further, it obtained a ROUGE-L score of 0.2436. Test results also suggest pretty high scores with respect to several other metrics such as BERTScore and Litepyramid among others.

Encouraged by the results, we aim at working towards further improvements to the proposed MLLTS scheme.

References

Raksha Agarwal and Niladri Chatterjee. 2022. Votesumm: A multi-document summarization scheme using influential nodes of multilayer weighted sentence network.

Abeer Alzuhair and Mohammed Al-Dhelaan. 2019. An approach for combining multiple weighting schemes and ranking methods in graph-based multi-document summarization. *IEEE Access*, 7:120375–120386.

Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117. Proceedings of the Seventh International World Wide Web Conference.

Günes Erkan and Dragomir R. Radev. 2011. Lexrank: Graph-based lexical centrality as salience in text summarization. *CoRR*, abs/1109.2128.

Jon M. Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632.

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

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Jorge Valverde Tohalino and Diego R. Amancio. 2017. Extractive multi-document summarization using multilayer networks. *CoRR*, abs/1711.02608.