Grenze International Journal of Engineering and Technology, June Issue

Improve Performance of Abstractive Text Summarization using Bidirectional LSTM with Content-based Attention

Dakshata Argade[1] and Dr. Vaishali Khairnar[2]
[1-2]IT, Terna Engg College, Nerul, Navi-Mumbai, India
Email:dkshtargade@gmail.com

***Abstract*— Easy access of the internet generating tremendous amount of data every day. The data is generated at high volume with high velocity and to know the essence of data automatic text summarization is needed. The purpose of research is to design Abstractive Text Summarizer using deep learning techniques. In this study, we model the summarization task as a Sequence-to-Sequence (Seq2Seq) problem. The paper experiments with encoder decoder based different variants of RNN such as vanilla RNN, Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). The problem with existing approach is that the context vector contains information from the last recent part of the sentence and forgets the relevance of the initial time step input. To solve this problem, the content-based attention mechanism is applied which uses cosine similarity to generate context vector at each time step. The CNN/Daily Mail dataset is used for the study. The performance of models is measured using the ROUGE score to select the ideal model for summarization. The experimental results show bidirectional LSTM combined with a content-based attention mechanism enhances summarization performance. The experimental data demonstrates that content-based attention outperforms over dot product based attention.**

***Index Terms*— Abstractive text summarization, Attention, RNN, LSTM**.

I. INTRODUCTION

The digitalization makes an enormous amount of information available on a daily basis but it is very difficult for humans to find relevant information in less time. So, there is a great need of automatic summarization tools in this information rich era. Text Summarization is the process of shortening of long documents by retaining its original information.

The text summarization is generally classified into two categories as extractive and abstractive text summarization [1]. The extractive text summarization means extracting important sentences from source documents to form a summary. The abstractive text summarization is resembles to the way humans summarize the document. Instead of paraphrasing the original text, the abstractive text summarization creates new sentences. A lot of research has been carried out on extractive summarization, but due to the complexity of abstractive summarization, it is not that much explored [1].

The power of deep learning proves its success in various fields. Deep learning is being used in machine translation, which inspires people to use it for other NLP issues [2].Text summarization is a Seq to Seq problem where the

Grenze ID: 01.GIJET.10.2.405

input document is a sequence of words and the output is also a sequence of words [5]. The recurrent neural network is giving promising results. Researchers started using RNN for summarization task, most of which used encoder decoder architecture which encodes the input into a context vector and the decoder decodes the context vector again into a sequence of data [4]. The RNN works well for shorter length sentences but as sentence length goes on increasing RNN faces problem of vanishing and exploding gradient. To Solve this problem, enhanced variants of RNN i.e. LSTM and GRU are developed [7,8]. The LSTM has a memory cell due to which it can remember context over long sentences. GRU is a simplified version of LSTM because of the less number of gates. GRU need less time in training than LSTM. In earlier work of summarization, unidirectional RNN used which captures information of one side i.e forward direction to predict the next word. To improve performance further, bidirectional LSTM with an attention mechanism has been used. The above discussed approaches are capable of capturing most recent information from input sentence and forgot past information. In summarization, recent as well as past information is also important. To solve the above problem, a content-based attention mechanism is proposed, which computes content-based similarity between source and target using cosine similarity. The main contribution of this work is as follows:

1. To design and analyze abstractive summarization using deep learning techniques in order to choose the best summarization method.
2. To customize attention mechanism with content-based similarity over dot Product based Attention.

The remaining portion of the manuscript is organized as follows:
The section 3 discussed about the literature review. The proposed methodology is explained in section 3. The findings and conclusion of this research described in section 4 and section 5 respectively.

II. Literature Review

Since deep learning makes it easier to learn multilevel hierarchical representations of data utilizing multiple nonlinear data processing layers, it is used in many NLP tasks [5-7]. Sequence-to-sequence models, often known as deep learning models, have recently gained popularity in NLP and have shown promise for solving problems including machine translation [11], speech recognition [11], and video captioning.
The variety of deep learning models, such as RNNs, convolutional neural networks (CNN) & seq-to-seq models, have been applied to abstractive summarization. In 2015, a model based on the encoder-decoder architecture was suggested for abstractive text summarization [8], introducing the first application of deep learning techniques. Deep learning algorithms have been employed extensively in recent years, generating excellent results for many applications. The seq-to-seq idea is the foundation of the RNN encoder decoder architecture. The seq-to-seq model translates the input sequence of the neural network into a matching sequence of letters, words, or phrases. Data is processed sequentially using a deep learning model known as an RNN where the outcome of one state influences the input of the next [9, 10].
Bidirectional RNN was employed in later research for text summarization. There are two RNNs i.e forward and backward RNNs in a bidirectional RNN. The forward RNN scans the input from left to right and generates a hidden state sequence. On the other hand, after reading the input sequence from right to left, the backward RNNs produce a sequence of hidden states. The forward and backward RNNs states are combined to represent the input sequence [11]. As a result, how each word is represented relies on how the words that come before it (the past) and after it (the future) are represented. This is due to the unidirectional RNN's exclusive consideration of the prior prediction and past-focused reasoning. The bidirectional RNN [12] can be used to solve this issue because if there is a mistake in one prediction then it will be carried in all subsequent predictions.
The attention mechanism was first used for machine translation task [11] followed by other NLP tasks such as summarization [8]. To understand the gist of input documents, RNN with an attention mechanism is included, which not only takes into account the essential source segments but also directs them throughout the decoding process. The input string encoding size is fixed and cannot take into account all of a long string's elements, a base architecture of encoder-decoder may not perform well when long phrases are given.
The attention mechanism was introduced to help people remember the information that significantly affects the summary [12]. The two RNN variations LSTM [13] and GRU [14] are the most widely used gated RNNs. By integrating a memory cell within their network, GRU and LSTM are able to store data from early in the sequence to be used later in the sequence, which solves the issue of vanishing gradients that standard RNN units suffer from.
LU Yang [16] solved the abstractive summarization problem using a feed-forward neural network and an attention-based encoder. In order to increase the likelihood of receiving a more relevant summary to the review, he initialized the vector, employed word embedding, and employed the beam search technique to generate predictions. Liu et

al. [15] develop a hybrid model that combines abstractive and extractive summarization techniques using the BERT word embedding and reinforcement learning techniques. Reinforcement learning is used to connect the generated abstraction and extraction networks. By taking into account both semantic and grammatical factors, the summary has demonstrated strong performance and outcomes when tested on the Daily Mail dataset.

III. PROPOSED MODEL

A. Bi-directional LSTM with Content-based Attention Mechanism

With the invention of deep neural network and its performance in machine translation problem, motivates all researchers for application of deep neutral network in field of NLP including task of summarization [3,6]. The problem of Text summarization can be formulated mathematically as mentioned below Given document D consists of many tokens D= (x_1, x_2, x_3...x_n) abstractive text summarization aims to generate shorter, concise relevant summary Y= (y_1, y_2, y_3...y_m) that catch important information from source document D and usually m<n/2. Every token is part of pre-defined fixed vocabulary V.

The proposed model is shown in Figure 1, there are three main components of our model.

1. Bidirectional Encoder
2. Attention Mechanism
3. Bidirectional Decoder

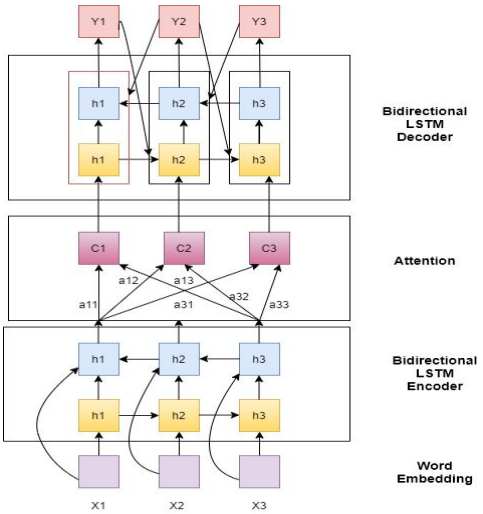

Figure.1 Abstractive Text Summarization System Architecture

B. Bidirectional Encoder-Decoder

Basically encoder and decoder architecture used to solve Seq-Seq problem where input is sequence and generated output is also sequence [12]. The encoder encodes the entire input into a representational fixed-length context vector, which is further passed to the decoder to decode information from it. The LSTM unit is used as an encoder and decoder. In this work, bidirectional LSTM is used to capture past and future information. It processes data in the forward and backward directions. To capture past and future information from a sequence of words, a bidirectional encoder is used.

Basically, for predicting the next possible word provided, the previous sequence of words fully depends on the previous words of the sentence, but it also depends on the future words of the sentence. In order to store both past and future data, we employed bidirectional long short-term memory [11]. The LSTM receives the word sequence as input in both forward and backward directions. The hidden state is calculated as per equation 1 and 2.

$$\overrightarrow{H_{tt}} = \overrightarrow{\text{LSTM}}\ (X_t, h_{t-1}) \qquad (1)$$

$$\overleftarrow{H_{tt}} = \overleftarrow{\text{LSTM}}\ (X_t, h_{t+1}) \qquad (2)$$

In this work, the decoder is also bidirectional and consists of two LSTMs: one decodes information from left to right and the second decodes information from right to left. Bidirectional LSTM takes time to train, but it improves

the performance of the model. At the Decoder end, the softmax function is used to predict the target token of the summary. The softmax function converts decoder output into a probability distribution over vocabulary |V| [11]. The probability is calculated based on the previous token and the decoder's hidden state.

C. Attention Module

In the encoder decoder architecture, the entire input is encoded into a fixed length context vector, which contains the gist of the entire input. The problem with this approach is that the context vector contains information from the last recent part of the sentence and forgets the relevance of the initial time step input. To solve this problem, an attention mechanism is used where, instead of generating a single generalized context vector from the encoder's last hidden state, it creates a context vector at each timestep [13].

D. Attention layer

The attention layer often uses a dot product-based attention algorithm. The attention mechanism is depicted in Figure 2. The Relu activation function is used to normalize the results once a dot product between each encoder and decoder output is obtained. According to intuition, the dot product indicates how closely the decoder and encoder outputs are related, rewarding related words and penalizing unrelated ones. Instead of remembering unnecessary words, this layer of attention made it easier for the seq-seq architecture to concentrate on terms that are relevant to the summary.

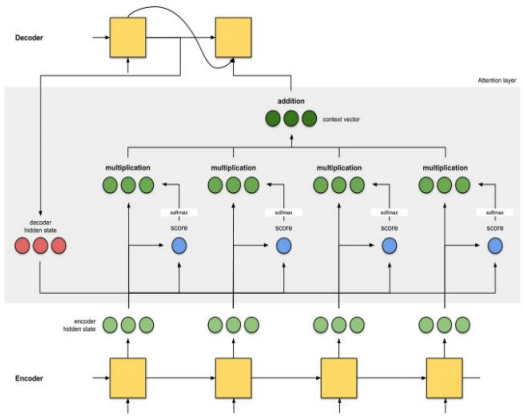

Figure. 2 Attention Layer. Source: Taken from [17]

E. Working of Attention Mechanism

In a neural network model, an attention mechanism usually consists of the following steps:

- *Input Encoding*: In this step, input data is encoded using the embedding to convert input into a fixed length vector. The attention mechanism takes this transformed input to process further.
- *Query Generation*: The model's context or current state is used to construct a query vector. The information that the model wants to extract or concentrate on from the input is represented by this query vector.
- *Key-Value Pair Creation*: Key-value pairs are created from the input representations. The values hold the actual data or information, while the keys hold the information that will be utilized to assess the relevance or importance.
- *Similarity Computation*: Each key's similarity to the query vector is calculated to determine how relevant or compatible they are. A variety of similarity metrics, including scaled dot product, cosine similarity, and dot product, can be applied.
- *Dot-product Attention*: In Dot-product, the attention weights are calculated by taking the dot product of the key vectors and the query.
- *Scaled dot-product Attention*: Dot-product scaling is a kind of attention that is based on the scaling of the dot product by the square root of the key dimension.
- *Attention Weights Calculation*: To determine attention weights, the similarity scores are run via a softmax function. Each key-value pair's weight denotes its significance or relevance. A weighted sum is produced by applying the attention weights to the respective values. This stage compiles the significant data from the input according to the attention mechanism's ranking of relevance.

- *Context Vector*: The context vector is a weighted sum of important and attended information from the input. It captures the main relevant content for the current step.
- *Integration with the Model*: The context vector is integrated with the hidden representation or current state of the model to give more context or information for the model's layers or upcoming stages.

F. Content-based Self Attention

Self-attention provides attention over the same sentence only to learn a better representation of itself. Cosine similarity is used to calculate the content-based attention scores, which are then normalized using softmax [31]. At each time step i in the source sentence, the encoder generates a hidden state h_i. The decoder generates hidden state s_j at each time step j. After that, the alignment score as_{ij} is computed, which shows which source word is aligned with the target word using a score. The hidden state of the encoder, i.e. source (h_i), and the hidden state of the decoder (target s_j) are used to calculate the alignment score. The score function used as cosine similarity function

$$as_{ij} = Cosine\_similarity\ (h_i, s_j) \quad (3)$$

The softmax function is used to normalize the alignment score.

$$a_{ij} = e^{as_{ij}} / \sum_{k=1}^{Tx} e^{e_{ik}} \quad (4)$$

To calculate the attended context vector, we find the sum of the product of attention weight and the hidden state of the encoder h_i.

$$C_i = \sum_{j=1}^{Tx} a_{ij}.h_j \quad (5)$$

The context vector C_i and hidden state of the decoder at time step i are combined to generate attended hidden vector S_i given as

$$S_i = concatenate(s_j, C_i) \quad (6)$$

The attended hidden vector S_i is then passed to the dense layer to compute Y_i. The softmax function gives probability distribution over vocabulary for target output.

IV. Experimental Results

In this work, we have implemented four different RNN's models as follows:

1. Long Short Term Memory
2. LSTM with Attention Mechanism
3. Gated Recurrent Unit
4. Bidirectional LSTM with Content-based Attention

We implemented our models in the Tensor Flow library. We employed stochastic gradient descent with 32-miniature batches to reduce our loss. As we train, the hyper-parameters are tuned based on the perplexity of summaries in the validation set. The various models that have been created are assessed using the Pyrouge library, which provides measures for precision, recall, and F-score.

A. Dataset

More than 300,000 news stories make up the CNN/Daily Mail dataset), each of which is coupled with a number of highlights known as multi-line summaries. According to their scripts, this corpus has a total of 286,817 training pairs 11,487 test pairs, and 13,368 validation pairs.

B. Parameter Settings

In all of our tests, the word embedding and hidden state dimensions for encoder as well as decoder are set to 128 and 256 respectively. The embeddings are completely learned during training. For stochastic optimization, the Adam optimizer with hyper-parameters $\beta1 = 0.9$, $\beta2 = 0.999$, and $\epsilon = 10{-}8$ is set. Mini-batches of size 16 are utilized, and the learning rate is fixed at 0.001. Gradient clipping is also employed when the gradient norm is no greater than 2.0. The vocabulary is shared across source and target for all datasets and has 50K words. We limit the length of summaries to 30 tokens and truncate source articles to 100 tokens. The model is trained using 10 epochs over the CNN/Daily Mail dataset. We chose a beam size of 5 for testing.

V. Results And Discussion

We have implemented abstractive text summarization using different RNN models. The rouge metric is selected for evaluating the performance of the developed models. It is observed that bidirectional LSTM obtained better rouge score so we select it as best model and customized it further with content based attention. We got a rouge score up to 0.48 through bidirectional LSTM during the testing of data on the CNN-Daily Mail dataset.
On the basis of perplexity on the held-out set, the models are evaluated. The Precision, Recall, and F-measure for ROUGE-1, ROUGE-2, and ROUGE-L are computed using the model that has the best perplexity, and all the results are reported. ROUGE known as Recall-Oriented Understudy for Gisting Evaluation. It is a set of measurements used to assess machine translation and text summarization that is done automatically. It compares machine-generated summaries with reference summaries created by humans. The term "recall" refers to the amount of the reference summary that the system summary is able to retrieve or record. The overlap of unigrams between the reference summary and system summary is provided by ROUGE-1 where, whereas ROUGE-2 describes the bigram overlap between reference summaries and the system. Table 1 shows the performances of all four models in terms of rouge score. The rouge score in terms of R-1, R-2 and R-L has been calculated for all the implemented models and reported in Table 1. The bidirectional LSTM model gives better performance than the rest of the three models.

Table I. Experimental Results of Models

Sr. No.	**Models**	**R-1 Precision**	**R-1 recall**	**R-1 fmeasure**	**R-2 precision**	**R-2 recall**	**R-2 fmeasure**	**R-L precision**	**R-L recall**	**R-L fmeasure**
1	LSTM	0.3714	0.4167	0.3811	0.1642	0.1818	0.1671	0.2503	0.2835	0.2579
2	LSTM with Attention	0.3824	0.4411	0.3985	0.1760	0.2013	0.1822	0.2658	0.3090	0.2775
3	GRU	0.3862	0.4703	0.4119	0.1828	0.2241	0.1952	0.2683	0.3308	0.2874
4	Bi-directional LSTM	0.3848	0.4848	0.4163	0.1823	0.2317	0.1978	0.2645	0.3371	0.2875

Table II. Comparison Between Cosine And Dot Product Attention

Rouge Metric	Models					
	LSTM		GRU		Bi-directional LSTM	
	Cosine Attention	Dot- Product Attention	Cosine Attention	Dot- Product Attention	Cosine Attention	Dot-Product Attention
R-1	0.4514	0.4411	0.4923	0.4703	0.4989	0.4848
R-2	0.2131	0.2013	0.2371	0.2241	0.2454	0.2317
R-L	0.3298	0.3090	0.3408	0.3308	0.3456	0.3371

Table II presents the results got in terms of rouge scores by the different RNN variants where the attention layer compares data using cosine similarity rather than dot product. A comparison is done between dot product and cosine attention, and the findings are observed. Performance improved with the use of cosine attention for generated summary. There was an increase in rouge scores because the cosine similarity computation significantly having greater accuracy than the vector's dot-product computation.

In figure 3, the obtained rouge score by four different summarization models is shown. The proposed model shows the highest rouge score among the four models. The Bi-directional LSTM with content-based similarity attention performs better.

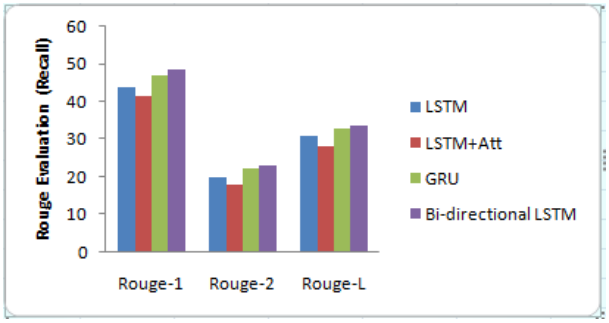

Figure.3. Performance (Recall-based Rouge) comparison on CNN/Daily Mail dataset.

As shown in Fig 4&5, we measure precision and F-measure for developed models which shows bidirectional model with content based attention mechanism give consistent performance with other models.

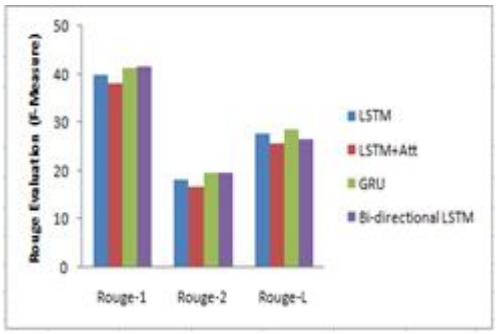

Figure.4. Performance (F-Measure) comparison on CNN/Daily

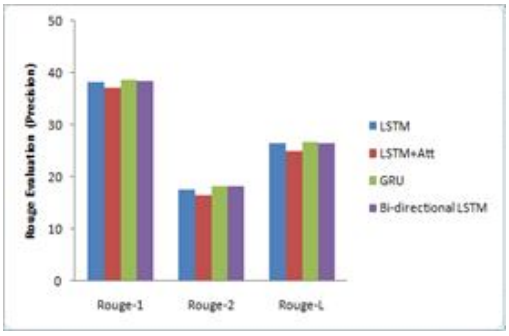

Figure5. Performance (Precision) comparison on CNN/Daily mail dataset

VI. CONCLUSION

We have implemented different models for abstractive summarization using Bidirectional LSTM Encoder-Decoder Architecture along with LSTM, LSTM with Attention and GRU models. The models are trained on the CNN corpus to generate a summary. Experimental results show that the Bidirectional LSTM model performs better than the other three models. We got a rouge1 score up to 0.48 during the testing of data on the CNN dataset. We customized the attention mechanism by adapting cosine similarity over dot product attention as the similarity measure. The emphasis to cosine similarity resulted in an increase in rouge scores. The content-based similarity provides better performance over dot product. In future, the Model can be improved further with pre-trained models.

REFERENCES

[1] Allahyari, Mehdi, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, and Krys Kochut.“Text summarization techniques: a brief survey." arXiv preprint arXiv:1707.02268 (2017).

[2] Khan, Atif, Naomie Salim, Haleem Farman, Murad Khan, Bilal Jan, Awais Ahmad, Imran Ahmed, and Anand Paul. "Abstractive text summarization based on improved semantic graph approach." International Journal of Parallel Programming 46, no. 5 (2018): 992-1016.

[3] Y. Jaafar and K. Bouzoubaa, “Towards a new hybrid approach for abstractive summarization,” Procedia Computer Science, vol. 142, pp. 286–293, 2018.

[4] Rush, Alexander M., Sumit Chopra, and Jason Weston. "A neural attention model for abstractive sentence summarization." arXivpreprint arXiv:1509.00685 (2015).

[5] Shi, Tian, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. "Neural abstractive text summarization with sequence-to-sequence models." ACM Transactions on Data Science 2, no. 1 (2021): 1-37.

[6] JJoshi, Akanksha, E. F. Fernández, and E. Alegre. "Deep learning based text summarization: approaches, databases and evaluation measures." In International Conference of Applications of Intelligent Systems. 2018.
[7] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning. nature 521 (7553), 436-444.
[8] Young, Tom, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. "Recent trends in deep learning based natural language processing." ieee Computational intelligenCe magazine 13, no. 3 (2018): 55-75.
[9] Song, Shengli, Haitao Huang, and Tongxiao Ruan. "Abstractive text summarization using LSTM-CNN based deep learning." Multimedia Tools and Applications 78, no. 1 (2019): 857-875.
[10] SSchuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." IEEE transactions on Signal Processing 45, no. 11 (1997): 2673-2681.
[11] HHochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9, no. 8 (1997): 1735-1780.
[12] Chopra, Sumit, Michael Auli, and Alexander M. Rush. "Abstractive sentence summarization with attentive recurrent neural networks." In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 93-98. 2016.
[13] YYao, Kaichun, Libo Zhang, Dawei Du, Tiejian Luo, Lili Tao, and Yanjun Wu. "Dual encoding for abstractive text summarization." IEEE transactions on cybernetics 50, no. 3 (2018): 985-996.
[14] K. Cho, “Learning phrase representations using RNN encoder–decoder for statistical machine translation,” in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734, Doha, Qatar, 2014.
[15] Wang, Q., Liu, P., Zhu, Z., Yin, H., Zhang, Q., and Zhang, L. (2019) ‘A Text Abstraction Summary Model Based on BERT Word Embedding and Reinforcement Learning’, Applied Sciences, vol. 9, no. 21, p. 4701.
[16] Yang, L. (2016) ‘Abstractive Summarization for Amazon Reviews’, Stanford University.
[17] ‘Attn: Illustrated Attention‘, towards data science, n.d. [Online]. Available: https://towardsdatascience.com/attn-illustrated-attention5ec4ad276ee3. [Accessed: April 10, 2021].