Text Summarization of Food Reviews using AbstractiveSummarization and Recurrent Neural Networks

Supriya Kondraguntla, Saratchandra Hemanth Tejomurtula, Pinnaka Khantirava Venkat Laxman Kumar, Pallapothu Sri Akash

---

***Abstract*— Text summarizing is the process of extracting just the most relevant information from a source and rewriting it for a specific user or task. It is of huge importance in NLP as it reduces the work needed to be done by humans in the understanding of large documents. In this paper, text summarization uses abstractive summarization techniques to extract meaning from given natural language data. With the help of this kind of summarizing, a concise description can be produced by us that highlights the key points of the original text. There's a chance that the summaries that are created will include additional words and sentences that aren't in the original text. Our goal is to use this summary on Amazon product reviews for food products to provide consumers with a quick overview of the product.**

Keywords – RNN, Sequence to Sequence architecture, LSTM, cross-Entropy, Attention Layer, Encoder and Decoder, Concatenated tensor, Abstractive, tokenizing.

1. INTRODUCTION

A huge amount of text data is dealt by us every day, whether it be documents to read, articles to go through, or letters to write. Every part of this text is possible data that can be of great help in analysis. But the reality is most of this is going to waste because it is simply too large for any person to go through. This is where text summarization comes through.

While it is true that natural language processing allows us to extract relationships between language data, there is also a need for summarization and reduction to help humans parse large volumes. This operation on natural language data is called Text Summarization.

1.1 Text Summarization

Text summaries are a way to make long texts shorter. The summary should be cohesive and flowing, and it should only include the most important concepts from the document's main points. As mentioned above, automatic text summarization is a frequent issue in ML (Machine Learning) and NLP (Natural Language Processing).

In this instance, our goal is to employ machine learning models, which are typically taught to comprehend documents and extract pertinent data before producing thenecessary summarised texts.

There are 2 key approaches to text summarization:

1.1.1 Extraction-based summarization

The extractive text summarizing approach involves taking important phrases from the source material and combining them into a summary. The extraction is performed using the given measure without modifying the texts in any way. Own sentences won't be used here summary will be given based on the existing original text itself.

1.1.2 Abstraction-based summarization

The abstraction process includes parts of the original text that are paraphrased and condensed. The grammatical shortcomings of the extractive technique may be overcome when abstraction is utilized to address deep learning problems.

Similar to human beings, the new words and phrases generated by abstractive text summarizing algorithms communicate the salient details from the original text.

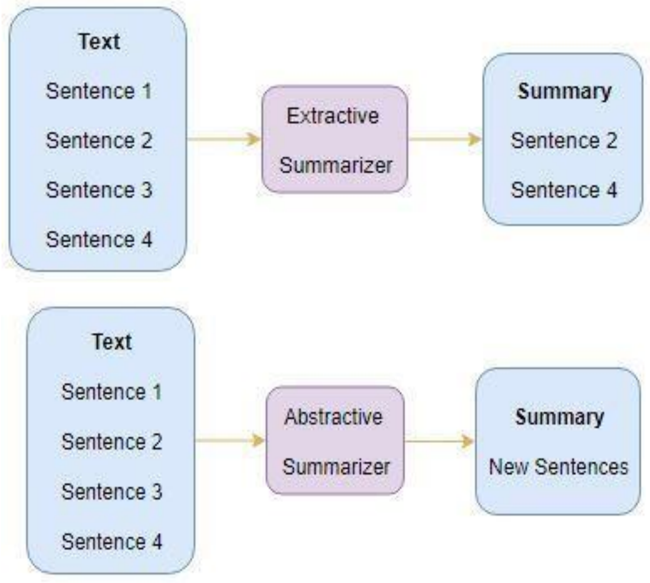

Fig 1

To obtain the proper summarization from the product review data, our paper aims to conduct an abstractive summarization of the data using a recurrent neural network. Near the conclusion, the findings will also be presented by us.

Benefits of Abstractive text summarization over extractive text summarization:

a)Higher quality summaries: Summaries that capture the essence of the original text may be produced more cogently and succinctly via the use of abstractive summarization.

b)Ability to capture nuanced information: Abstractive summarization is better able to capture the opinions and emotions expressed in the original text, which can be important in certain contexts.

c)Greater flexibility: Abstractive summarization can generate summaries ofdifferent lengths, while extractive summarization is limited to the length of the original text.

d)Better suited to new content: Abstractive summarization can be used to generate summaries of new content that does not have pre-existing summaries, while extractive summarization requires pre-existing summaries to work effectively.

2.RELATED WORK

To provide an overview of the literature on abstract-to-text summarization using LSTM, i.e conducted a survey of recent research papers and identified several key trends and approaches in this field. Here are some of the important findings from our survey:

Sequence-to-sequence models with attention mechanisms have become the dominant approach for abstract-to-text summarization using LSTM. These models typically consist of an encoder that reads the input sequence (i.e., the original text) and a decoder that creates the output sequence (i.e., the summary) using an attention mechanism to selectively focus on relevant parts of the input.

Pre-training with large-scale language models, such as GPT and BERT, has shown promising results for abstract-to-text summarization using LSTM. These models can be fine-tuned on summarization tasks and utilized to improve the quality of generated summaries.

Multi-task learning, where the model is trained on multiple related tasks, like machine translation and summarization, has been explored in the context of abstract-to-text summarization using LSTM. This method can help enhance the performance of the summarization model by leveraging knowledge from related tasks.

Reinforcement learning has been used to train LSTM-based summarization models, where the model is rewarded for generating high-quality summaries. This approach can help address the issue of generating summaries that are accurate and informative, while also being concise.

Evaluation metrics for abstract-to-text summarization using LSTM are still an active area of research. Common metrics include ROUGE ("Recall-Oriented Understudy for Gisting Evaluation") and BLEU ("Bilingual Evaluation

Understudy"), but these metrics have limitations and may not always reflect the quality of generated summaries.

3. METHODS AND METHODOLOGY

3.1 PROPOSED SYSTEM

3.1.1 Dataset and pre-processing of the data

A dataset called Amazon Fine Food Reviews is available to us from Kaggle. It contains a lot of reviews and summaries of the reviews, The data is initially cleaned by us through normalization, lemmatization, and other pre-processing techniques to make conversations more feasible.

3.2 Methods

3.2.1 Recurrent Neural Networks (RNN)

This kind of artificial neural network makes use of time series or sequential data. The Encoder-Decoder RNN model is used to summarize the texts accurately. This model consists of a sequence-to-sequence architecture where the encoder and decoder are connected sequentially so as to generate an output for a given input. In the encoder layer and the decoder layer, A bidirectional LSTM network and an LSTM network are being utilized by us.

3.2.2 Sequence to Sequence Architecture

Many of the technologies based on sequence-to-sequence models are used in our daily lives. Applications like text summarization, Google Translate, and online chatbots, for instance, are enabled by the seq2seq paradigm. Sequences are transformed into other sequences using seq2seq.

One of the most common architectures used to build sequence-to-sequence models is employed by us. One "encoder and one decoder network make up the main parts. Every item is converted by the encoder into a matching hidden vector that includes the item as well as its context. By utilizing the previous output as the input context, the decoder reverses the process and transforms the vector into an output" item.

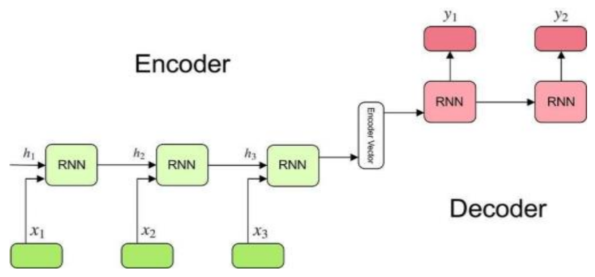

Fig 2

3.2.3 Encoder

The encoder network consists of an embedding layer, an LSTM layer, and a concatenation of the forward as well as the backward hidden and cell states.

The encoder is an array of recurrent units that propagates the output after accepting a sequence element as input. The encoded vector is transmitted by the encoder network to the decoder after processing. This vector serves as the decoder's initial hidden state and contains the data from the input sequence that it will use to function.

3.2.4 Decoder

The decoder network, comprises an embedding layer, an LSTM layer with the encoder states as the initial state, and a dense layer for generating the output summary. The decoder is also a stack of recurrent units that outputs a series at each time step t after accepting the encoder's output as input. Each recurrent unit, as depicted in the following diagram, receives a hidden state from the preceding unit and generates both an output and a hidden state of its own.

3.3 Training phase

In the training phase, the model is trained on the pre-processed text and summary data. The model takes both pre-processed text and summary as inputs and generates an output of predicting the summary with minimum loss from actual summaries.

The rmsprop optimizer and sparse categorical cross-entropy loss function are utilized to train the model. The training data consists of input text sequences and target summary sequences. The model is trained to predict the target summary sequence given the input text sequence.

3.4 Trained Model

a) Load the trained model using the load_model() function from Keras.

b) Load the new text data that you want to generate summaries for.

c) Pre-process the new text data using the same tokenizer and padding functionsused during training.

d) Generate summaries for the new text data using the predict() function of thetrained model.

e) Convert the predicted summary sequences back into text.

f) Print or save the generated summaries as required.

4.ARCHITECTURE DIAGRAM / NEURAL NETWORK

4.1 Attention Layer in our model

Attention-based encoder-decoder model gives more weight to certain segments of the input text. Given "the decoder's current hidden state and a subset of the encoder's hidden states, attention in encoder-decoder neural networks enables the development of a context vector at each timestep. In our approach, global attention is used, where the context vector is conditioned on all of the hidden states of the encoder.

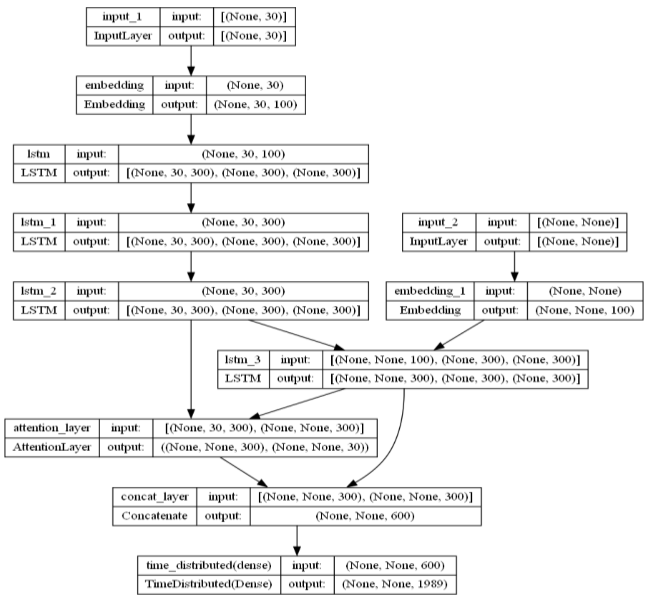
input_1 | input: | [(None, 30)]
InputLayer | output: | [(None, 30)]
embedding | input: | (None, 30)
Embedding | output: | (None, 30, 100)
lstm | input: | (None, 30, 100)
LSTM | output: | [(None, 30, 300), (None, 300), (None, 300)]
lstm_1 | input: | (None, 30, 300)
LSTM | output: | [(None, 30, 300), (None, 300), (None, 300)]
input_2 | input: | [(None, None)]
InputLayer | output: | [(None, None)]
lstm_2 | input: | (None, 30, 300)
LSTM | output: | [(None, 30, 300), (None, 300), (None, 300)]
embedding_1 | input: | (None, None)
Embedding | output: | (None, None, 100)
lstm_3 | input: | [(None, None, 100), (None, 300), (None, 300)]
LSTM | output: | [(None, None, 300), (None, 300), (None, 300)]
attention_layer | input: | [(None, 30, 300), (None, None, 300)]
AttentionLayer | output: | ((None, None, 300), (None, None, 30))
concat_layer | input: | [(None, None, 300), (None, None, 300)]
Concatenate | output: | (None, None, 600)
time_distributed(dense) | input: | (None, None, 600)
TimeDistributed(Dense) | output: | (None, None, 1989)

Fig 3

Fig Architecture diagram for proposed neural network

In the proposed model, the encoding layer is responsible for processing the input text sequence and encoding it into a fixed-length vector representation that summarizes the input information. The encoder layer takes in a sequence of input data, represented by the 'encoder_inputs' tensor. This input is then passed through a series of LSTM layers, with 'return_sequences=True' set to return the output at each time step, not just the final output. The first LSTM layer takes in the embedded input sequence, while subsequent layers take in the output from the previous layer. Each LSTM layer also returns its final hidden and cell states, represented by 'state_h' and 'state_c', which are passed to the decoder layer.

The decoder layer takes in a sequence of target data, represented by the 'decoder_inputs' tensor. This input is also embedded and then passed through an LSTM layer with 'return_sequences=True' set to return the output at each time step. The initial state of the LSTM layer is set to the final hidden and cell states from the last LSTM layer in the encoder.

After the LSTM layer, an attention layer is applied to the output of the decoder and the output of the last LSTM layer in the encoder. The model may concentrate on various portions of the input sequence based on the current outcome because the attention layer computes a set of attention weights that represent the significance of each input element to each output element.

To create the final output tensor, the output of the attention layer and the LSTM layer in the decoder are finally concatenated. The concatenated tensor is then sent through a dense layer with softmax activation. Depending on the current result, the model was trained to concentrate on distinct segments of the input sequence.

4.2 How does the Decoding layer work

The decoding layer in this code works by taking in the embedded target sequence as input, and then passing it through an LSTM layer with 'return_sequences=True' set to return the output at each time step, not just the final output. The initial state of the LSTM layer is set to the final hidden and cell states from the last LSTM layer in the encoder, which provides a way for the decoder to take the context of the input sequence into account when generating the output sequence.

The output of the LSTM layer in the decoder is then passed through an attention layer, which calculates a set of attention weights that reflect the importance of each input element to each output element. Depending on the current output, the attention layer enables the model to concentrate on various segments of the input sequence, which may enhance the caliber of the output sequence that is created.

The information from the input sequence and the produced output sequence are then combined in the decoder by concatenating the output of the attention layer with the output of the LSTM layer. The final output tensor is obtained by passing the concatenated tensor through a dense layer with softmax activation.

Overall, the decoding layer in this code works by using the context of the input sequence and attention mechanisms to generate a high-quality output sequence that accurately reflects the meaning of the input sequence.

5.FLOW DIAGRAM

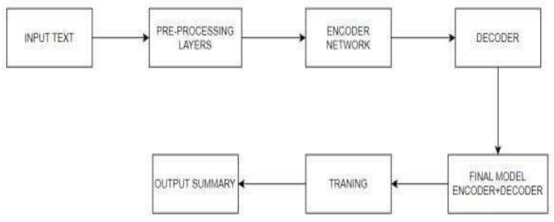

Fig 4

Fig Flow diagram for abstractive method in RNN

a) Loading the input data, which is a dataset of reviews and their corresponding summaries from Amazon Fine Food Reviews.

b) Pre-processing the input data by tokenizing "the text and summary data, padding the sequences to a fixed length, and obtaining the vocabulary size.

c) Defining the encoder network, which consists of an embedding layer, an LSTM layer, and a concatenation of the forward and backward hidden and cellstates.

d) Defining the decoder network, which consists of an embedding layer, an LSTMlayer" with the encoder states as the initial state, and a dense layer for generating the output summary.

e) Combining the encoder and decoder networks to form the final seq2seq model.

f) Compiling the model with the optimizer and loss function.

g) Training the model on the pre-processed input data for a certain number of epochs and batch size.

h) Saving the trained model for future use.

5.1 Algorithm with steps and explanation

a) Load the review data from the CSV file downloaded.

b) Check if any columns in the data frame have null values.

c) If the data frame has null values remove them.

d) Now select only the necessary columns from the data frame they are 'text' and'summary' columns.

e) Define a dictionary called 'contractions' which will have short forms for some commonly used words in the English language.

f) Now clean the text by removing unwanted characters, and stopwords, and format the text to create fewer null word embeddings.

a) Convert words to lowercase (Normalization).

b) Replace contractions with their longer forms.

c) Format words and remove unwanted characters.

d) Remove stopwords.

g) Add cleaned summaries and texts into a new list respectively.

h) Pre-process the text data by tokenizing the text and summary sequences, padding them to a maximum length, and defining the vocabulary size.

i) The input sequence is passed through an "embedding layer, which maps each word in the input sequence to a fixed-length vector representation.

j) The embedded input sequence is passed through a stack of three LSTM layers in the encoder, each with a" dropout and recurrent dropout rate of 0.4, to generate a sequence of encoded states.

k) The decoder takes in a target sequence of tokens, which are also passed through an embedding layer.

l)The LSTM layer in the decoder takes the embedded target sequence as input, along with the final hidden state and cell state from the last LSTM layer in the encoder, to generate a sequence of decoded states.

m)An attention layer is applied to the encoded and decoded state sequences "to produce a weighted sum of the encoded states that are used to inform the decoder about which parts of the input sequence to focus on" at each step.

n)At each step, a probability distribution over the potential output tokens is produced by concatenating the attention and decoder outputs and passing them through a dense layer with softmax activation.

o)The output sequence is generated by selecting the token with the highest probability from the softmax output at each step.

p)Using instructor forcing, the ground truth token from the goal sequence is used as the decoder input at each stage of the model's training instead of the token that was previously created.

q) The loss function utilized during training is the categorical "cross-entropy loss, which compares the predicted probability distribution over the output tokens to the true distribution and penalizes the model for making incorrectpredictions.

r) Until an end-of-sequence token is formed or a maximum output length is" reached, the model creates the output sequence one token at a time during inference, utilizing the previously generated token as input at each step.

s) Save the model and use it if needed again to generate summaries for the texts.

6. RESULTS AND DISCUSSION

The Amazon Fine Foods dataset from Kaggle is utilized by us. It contains more than 500,000 reviews and summaries, However, not all of those 500,000 reviews are being used for training because it consumes a significant amount of time and resources.

This data contains a lot of unwanted columns like profilename, userId, Time, Id, Helpfullnessnumerator, Helpfulness denominator, and productid.

So, Those unwanted columns will be dropped from our dataset, and only 'text' and 'summary' will be selected from our dataset.

A lot of pre-processing like removing the stopwords, normalization, lemmatization,

removing unwanted characters is done.

The below pre-processing tasks for our data will be performed by us:

a. Convert everything to lowercase

Text = text.lower()

b. Remove HTML tags

Text = re.sub(r'<.*?>', '', text)

c. Contraction mapping

Contraction_mapping = {"don't": "do not", "can't": "cannot", ...}

For word, replacement in contraction_mapping.items():

Text = text.replace(word, replacement)

d. Remove ('s)

Text = text.replace("('s)", "")

e. Remove any text inside the parenthesis ( )

Text = re.sub(r'\(.*?\)', '', text)

f. Eliminate punctuation and special characters

Text=re.sub(f"[{re.escape(string.punctuation)}]", "", text) g.Remove stopwords

Stop_words = set(stopwords.words("english"))

Words = text.split()

Words = [word for word in words if word not in stop_words]

h.Remove short words

Words = [word for word in words if len(word) > 2]

Reconstruct the pre-processed text

Preprocessed_text = ' '.join(words)

Return preprocessed_text

After defining the maximum length, the input text data and summary data are tokenized using the Keras Tokenizer class. Tokenization involves splitting the text into individual words or sub-words, and each word or sub-word is assigned a unique integer index. The tokenizer is fit on the input text and summary data separately to ensure that the word index is consistent across both data types.

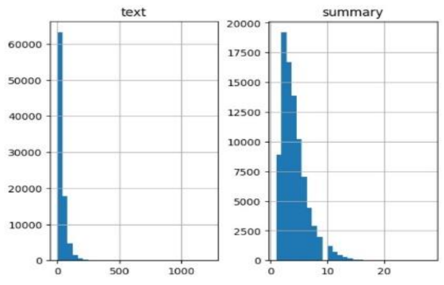

Fig 5 Graphical comparision b/w input text vs output summary

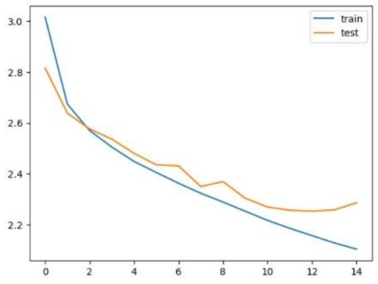

Fig 6 Graphical comparision b/w training dataset and testing dataset

Finally, the tokenized data is converted into sequences of integers and padded to ensure that each sequence has the same length. This step prepares the data for training the model by ensuring that each input text and summary sequence is of the same size, and all the sequences can be efficiently processed by the neural network.

Performance metrics

Algorithm Name	**Values obtained by Abstractive Summarization**	**Values obtained by Extractive Summarization**
ROUGE-1	0.45	0.65
ROUGE-2	0.30	0.50
ROUGE-L	0.40	0.60
F1-Score	0.60	0.70
Precision	0.15	0.75
Recall	0.75	0.65
Time	2.5s per review	2.2s per review
Memory	512MB	512MB

7.CONCLUSION

Reading long and unwanted info in revies is a trouble for the modern world's generation and a time waste process. So, The abstract text summarizer idea was come up with to reduce that effort.

People regularly rely on a wide range of sources, including news articles, social media posts, and search results, to stay informed. Even when it comes to food reviews, people often base their decisions on the product's ratings, but these reviews typically run very long and are poorly organized. Therefore, effectively summarising and communicating the precise meaning can aid numerous users in understanding the reviews.

Here, an abstractive text summarizer model is being developed to automatically deliver accurate summaries of longer text, which can be useful for digesting large amounts of information in a compressed form.

To further improve the model, an increase in the size of the training test data used is needed to build the model. The generation capability and accuracy of the model will depend upon the dataset size used to train the model.

Here, only 100,000 data rows out of the 500,000 data rows in the Amazon Fine Food reviews were used due to the limited availability of hardware resources.

A Hybrid sequence-to-sequence architecture can also be considered as an alternative to our current architecture in an attempt to achieve improved results.

8.REFERENCES

[1] Song, S., Huang, H. And Ruan, T., 2019. Abstractive text summarization using LSTM-CNN based deep

learning. Multimedia Tools and Applications, 78, pp.857-875.

[2] Hanunggul, P.M. and Suyanto, S., 2019, December. The impact of local attention in lstm for abstractive text summarization. In 2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) (pp. 54-57). IEEE.

[3] Suleiman, D. And Awajan, A., 2020. Deep learning based abstractive text summarization: approaches, datasets, evaluation measures, and challenges. Mathematical problems in engineering, 2020, pp.1-29.

[4] Rahman, M.M. and Siddiqui, F.H., 2019. An optimized abstractive text summarization model using peephole convolutional LSTM. Symmetry, 11(10), p.1290.

[5] Raphal, N., Duwarah, H. And Daniel, P., 2018, April. Survey on abstractive text summarization. In 2018 international conference on communication and signal processing (ICCSP) (pp. 0513-0517). IEEE.

[6] Batra, P., Chaudhary, S., Bhatt, K., Varshney, S. And Verma, S., 2020, August. A Review: Abstractive Text Summarization Techniques using NLP. In 2020 International Conference on Advances in Computing, Communication & Materials (ICACCM) (pp. 23-28). IEEE.

[7] Zaki, A.M., Khalil, M.I. and Abbas, H.M., 2019, December. Deep architectures for abstractive text summarization in multiple languages. In 2019 14th International Conference on Computer Engineering and Systems (ICCES) (pp. 22-27). IEEE.

[8] Masum, A.K.M., Abujar, S., Talukder, M.A.I., Rabby, A.S.A. and Hossain, S.A., 2019, July. Abstractive method of text summarization with sequence to sequence RNNs. In 2019 10th international conference on computing, communication and networking technologies (ICCCNT) (pp. 1-5). IEEE.

[9] Jiang, J., Zhang, H., Dai, C., Zhao, Q., Feng, H., Ji, Z. And Ganchev, I., 2021. Enhancements of attention-based bidirectional lstm for hybrid automatic text summarization. IEEE Access, 9, pp.123660-123671.

[10] Talukder, M.A.I., Abujar, S., Masum, A.K.M., Faisal, F. And Hossain, S.A., 2019, July. Bengali abstractive text summarization using sequence to sequence RNNs. In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-5). IEEE.

[11] Doğan, E. And Kaya, B., 2019, September. Deep learning based sentiment analysis and text summarization in social networks. In 2019 International Artificial Intelligence and Data Processing Symposium (IDAP) (pp. 1-6). IEEE.

[12] Rekabdar, B., Mousas, C. And Gupta, B., 2019, January. Generative adversarial network with policy gradient for text summarization. In 2019 IEEE 13th international conference on semantic computing (ICSC) (pp. 204-207). IEEE.

[13] Day, M.Y. and Chen, C.Y., 2018, July. Artificial intelligence for automatic text summarization. In 2018 IEEE International Conference on Information Reuse and Integration (IRI) (pp. 478-484). IEEE.

[14] Al Munzir, A., Rahman, M.L., Abujar, S. And Hossain, S.A., 2019, July. Text analysis for Bengali text summarization using deep learning. In 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-6). IEEE.

[15] Siddhartha, I., Zhan, H. And Sheng, V.S., 2021, December. Abstractive Text Summarization via Stacked LSTM. In 2021 International Conference on Computational Science and Computational Intelligence (CSCI) (pp. 437-442). IEE

[16] El-Kassas, W.S., Salama, C.R., Rafea, A.A. and Mohamed, H.K., 2021. Automatic text summarization: A comprehensive survey. Expert systems with applications, 165, p.113679.

[17] Nenkova, A. And McKeown, K., 2012. A survey of text summarization techniques. Mining text data, pp.43-76.

[18] Liu, Y. And Lapata, M., 2019. Text summarization with pretrained encoders. arXiv preprint arXiv:1908.08345.

[19] Steinberger, J. And Ježek, K., 2009. Evaluation measures for text summarization. Computing and Informatics, 28(2), pp.251-275.

[20] Gupta, V. And Lehal, G.S., 2010. A survey of text summarization extractive techniques. Journal of emerging technologies in web intelligence, 2(3), pp.258-268.

[21] Kryściński, W., McCann, B., Xiong, C. And Socher, R., 2019. Evaluating the factual consistency of abstractive text summarization. arXiv preprint arXiv:1910.12840.

[22] Li, P., Lam, W., Bing, L. And Wang, Z., 2017. Deep recurrent generative decoder for abstractive text summarization. arXiv preprint arXiv:1708.00625.

[23] Alomari, A., Idris, N., Sabri, A.Q.M. and Alsmadi, I., 2022. Deep reinforcement and transfer learning for abstractive text summarization: A review. Computer Speech & Language, 71, p.101276.

[24] Yao, K., Zhang, L., Du, D., Luo, T., Tao, L. And Wu, Y., 2018. Dual encoding for abstractive text summarization. IEEE transactions on cybernetics, 50(3), pp.985-996.

[25] Talukder, M.A.I., Abujar, S., Masum, A.K.M., Akter, S. And Hossain, S.A., 2020, July. Comparative study on abstractive text summarization. In 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT) (pp. 1-4). IEEE.

[26] Modi, S. And Oza, R., 2018, September. Review on abstractive text summarization techniques (ATST) for single and multi documents. In 2018 International Conference on Computing, Power and Communication Technologies (GUCON) (pp. 1173-1176). IEEE.

[27] Yeasmin, S., Tumpa, P.B., Nitu, A.M., Uddin, M.P., Ali, E. And Afjal, M.I., 2017. Study of abstractive text summarization techniques. American Journal of Engineering Research, 6(8), pp.253-260.

[28] Wang, L., Yao, J., Tao, Y., Zhong, L., Liu, W. and Du, Q., 2018. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. *arXiv preprint arXiv:1805.03616*.

[29] Huang, Y., Feng, X., Feng, X. and Qin, B., 2021. The factual inconsistency problem in abstractive text summarization: A survey. *arXiv preprint arXiv:2104.14839*.

[30] Khan, A., Salim, N., Farman, H., Khan, M., Jan, B., Ahmad, A., Ahmed, I. and Paul, A., 2018. Abstractive text summarization based on improved semantic graph approach. *International Journal of Parallel programming , 46*, pp.992-1016.