Abstractive Text Summarization

Project Final Report -2022

Muhammad Hamzah, mhamzah@andrew.cmu.edu
Raghad Abunabaa, rmabunab@andrew.cmu.edu

Abstract

The remarkable expansion of the volume of data available to us has raised the demand for automatic summarization. Very few studies focused on abstractive text summarization due to its complexity. In this paper, we build two neural networks models, more specifically a unidirectional long short-term memory and a hybrid long short-term memory which combines a bidirectional encoder and a unidirectional decoder. The aim of the study is to examine the effectiveness of long short-term memories to generate high quality summaries when compared with other models. The results show the hybrid long short-term memory is slightly more effective in generating meaningful summaries than the unidirectional long short-term memory. However, the model performed poorly compared to other richer long short-term memory models.

1 Introduction

The remarkable expansion of the volume of data available to us has raised the demand for automatic summarization to locate succinct and relevant information with the least amount of work and within the shortest time possible, which technical breakthroughs have made possible. The World Wide Web has significantly aided the growth of information by connecting millions of computers, producing a network in which individuals may exchange information anywhere, at any time, and faster and smoother than ever. Books, news, articles, blogs, and reviews are among the various sources of information that many people are interested in summarizing. Automatic summarization facilitates the selecting process for students, employees, managers, and others, permitting them to complete their tasks more efficiently.

Automatic summarization is the process of reducing overall text content while retaining meaning. The two most common strategies for summarizing text are extractive and abstractive summarization. Extractive summarization is the process of capturing the most important sentences from a text using statistical characteristics and then arranging the retrieved phrases to form the summary. In contrast, abstractive summarization develops when the summary is constructed using distinct phrases, either by rephrasing or by utilizing new terms, rather than merely extracting essential sentences (Gupta et al., 2019, p. 1).

There has already been substantial literature in the domain of automatic summarization as the curiosity in this discipline has peaked among many researchers with the various methodologies of creating text summarization models. Because of its ease of implementation, extractive summarization has been the topic among many studies. Scholars' interest in extractive summarizing has lately switched to abstractive summarization, since they have attained the full potential in extractive summarization models performance. Abstractive summarizing has proved particularly beneficial in lowering sentence length as it employs unification to integrate sentences, producing higher quality summaries with greater accuracy than extractive summarization. In this project, we will investigate how effective are long short-term memory model is in producing high quality summaries compared to other models.

2 Related Work

Abstractive text summarization has gained interest among many scholars as a result of the growing information and the need to read information within the shortest time possible. This section will provide an in-depth examination of the various approaches employed in prior research to develop abstractive text summarization models:

In a recent study by Cai et al. (2019), they proposed to use Relation Classification - Transformer (RC-T) to model abstractive summarization which includes an extra encoder with a bidirectional RNN to represent sequential context. It also includes a convolution module to capture local relevance as used in the previous study. RC-T can be trained to long-term interconnections and alleviate Transformer's fundamental flaw of being insensitive to word order information (p. 514). They used the Transformer as their baseline, which is aimed at solving sequence-to-sequence problems. The suggested model is made up of an RC encoder that converts the original text into a series of hidden states, allowing the model to capture the sequential context representation. It also incorporates a local convolution for learning n-gram features of varying sizes. It is also utilized to efficiently extract relevant information by filtering the sequential context with local relevance. Finally, these two representations are input into the decoder which outputs the summaries (p. 516). However, this model seems to be effective for short text.

In another study conducted by Nallapati et al. (2016), they employed an Attentional EncoderDecoder Recurrent Neural Network (RNN) to mimic abstract text summarization. They proceeded with the basic Encoder Decoder RNN with attention mechanism as their baseline. The encoder was a bidirectional Gated Recurrent Unit (GRU)-RNN, whereas the decoder was a unidirectional Contextual Recurrent Unit (CRU)-RNN. In the study, they have presented a variety of models that solve fundamental challenges in summarization that are reflected by the baseline model. For instance, they advocated using Feature-Rich Encoder to capture the keywords that revolve around the text using Feature-Rich Encoder which is an extra embedding matrix for the vocabulary of each tag-type: Part-Of-Speech (POS) and Named Entity Recognition (NER) tags. Moreover, they transform Term Frequency (TF) and Inverse Document Frequency (IDF) values into discrete values. which are merged with word embedding as input to the encoder (p. 2). In addition, they have proposed to use Switching Generator-Pointer to model unseen or uncommon words in which the decoder has a ‘switch’ that chooses whether to use the generator or the pointer at each time step. Finally, they employed a hierarchical encoder with hierarchical attention to capture hierarchical document structure along with the keywords using a bi-directional RNN at both the word and sentence level (p. 3). However, these models focused on producing summaries consisting of a single line.

Similarly, Song et al. (2018) also aimed to model abstract text summarization using a combination of two neural networks models, more specifically Long Short-Term Memory and Convolutional Neural Network (CNN). They proposed an LSTM-CNN based on Abstractive Text Summarization (ATS) framework that will generate new sentences by examining finer-grained pieces rather than sentences, particularly, semantic phrases. This was mainly through extracting phrases from the source text and then generating text summaries using deep learning. They used a Sequence-to-Sequence model consisting of an encoder and a decoder, both of which are RNNs, as their baseline. The proposed model takes phrases rather than words as input and output sequences. However, before feeding the phrases into the model, they employed the phrase extraction approach to break down sentences in the source text into phrase sequences that are then fed into the convolutional phrase encoder (p. 2). In addition, the recurrent document encoder was used to encode documents without limiting encoders' architecture. The output of the LSTM is then fed into the decoder which consists of two parts: generate mode which predicts the next phrase given the annotation and the copy mode where it copies the next phrase of the generated phrase as it doesn't fit the summary (p.3). However, training deep learning models is time consuming, making it inefficient for summarizing large documents.

As there are still numerous areas of development for the models provided, abstractive text summarization continues to capture the curiosity of many scholars. As a result, we will recreate abstractive text summarization using long short-term memory, which is one of the most extensively utilized strategies for abstractive text summarization by other researchers. In contrast to the above-mentioned models, we have chosen to build a unidirectional long short-term memory model that is trained on a new dataset and compare it against other models. This study has the potential to set the groundwork for future applications of abstractive text summarization.

3 Data Collection and Analysis

We trained and tested our abstractive summarization model on news articles which were collected by Akashsdas. The dataset originally consisted of 4,515 examples. However, it has been recently updated to include more than 12,000 examples. The data contains the following information: name of the author, headlines, url of the article, short text, and complete article. However, for the purpose of this study, we are only concerned with the complete article and the headlines which act as our short summaries. The data was scraped from news articles including Hindu, Indian times and Guardian over the period Feb - Aug 2017. The data was mainly collected for generating the headlines from the text. In other words, it was collected to summarize the large amount of information into a compressed space. We splitted the dataset into two files: training and testing files. The training dataset was 90% of the total dataset whereas the testing dataset was only 10%. This is to maximize the ability of our model to learn the features of the summaries in order to boost the performance of the model.

After reviewing the data, we believe that all summaries or the headlines are valid, but doesn't fully represent the text. We believe that this dataset will best serve the purpose of our study considering the fact that we are building simple long short-term memory models.

As a next step in developing our model, we have chosen the following preprocessing techniques on the three files to improve the processing of data:

- **Lowercasing:** Two identical terms will be regarded as unique when one is capitalized and the other is not. This would affect the size of the vocabulary, the frequency of words emerging in the datasets, hence affecting the dimensionality of matrices. This would increase redundancy and lowers the chance of a word becoming the next word, affecting the model's performance. Thus, it is necessary to lowercase all words in an effort to maximize the model's accuracy.

- **Expand Contractions:** As computer systems are clueless that contractions are acronyms for a series of words with the same meaning, two similar words will be considered to be entirely distinct things. Consequently, vocabulary size, word frequency and matrices' dimensions will all be affected, lowering the accuracy of the model. Furthermore, the elimination of punctuation marks would have an impact on contraction words, resulting in the emergence of new words that are not valid. For example, 'doesn't' would become 'does' and 't' if the punctuation was removed, reducing the model's accuracy. As a result, it is critical to extend contractions in order to enhance the model's accuracy.

- **Remove Punctuation:** As most of the punctuation marks follow a word without a space, it is expected that same words will be treated differently, again affecting vocabulary size, word occurrences and matrices' dimensions. Furthermore, Ek et al. (2020) reported that many neural networks, with the exception of BERT, exhibit a considerable decline in performance when there is a discrepancy in the presence of punctuation between the training and testing sets (p. 114). Hence, we removed punctuation marks to boost the performance of our proposed model.

- **Remove Stop Words:** Stop words are words that do not add value to the phrase. Nonetheless , they are generally the most prevalent terms in any dataset. Thus, deleting stop words would considerably minimize the time and space required to build the model. As a result, we omitted the stop words in order to increase the model's performance.

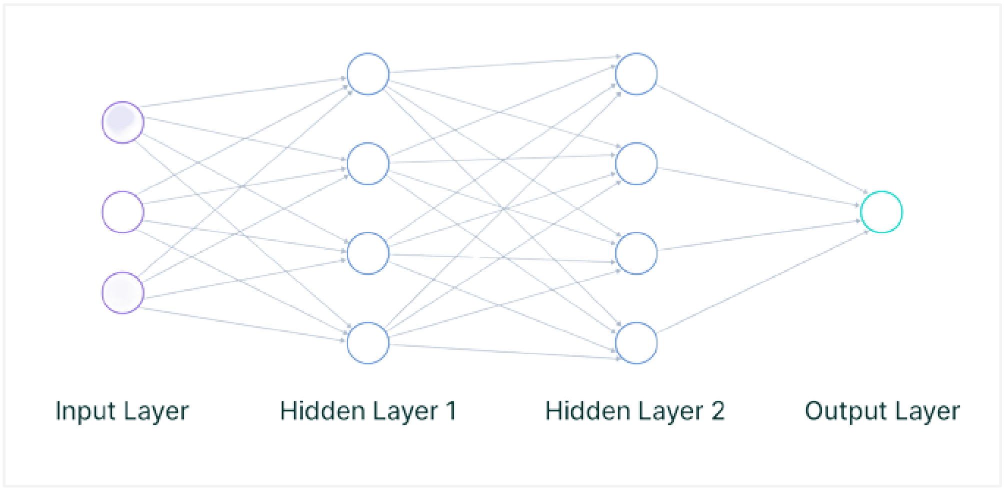
4 Methodology and Model Description

To create our abstractive summarization model, we will primarily employ Long Short-Term Memory (LSTM). Recent research has demonstrated that LSTMs outperform RNNs. This is because LSTMs address the issue of vanishing gradient, which happens when the gradient values are too small and the model stops learning, affecting the model's performance. Alternatively, the model would take too long to run, lowering the efficiency of our model. Furthermore, LSTMs are more effective for abstractive summarization because they eliminate the problem of information decay which arises with RNNs. This indicates that the model will be more suited to evaluate long-term dependencies, which become vital for obtaining meaningful information over long distances in the case of longer text. Thus, we developed a simple LSTM model as our baseline, and evolved it into a hybrid LSTM model which we expect to deliver better results as will be explained in the following section. In addition, we will also evaluate our model against more featured LSTMs such as X and Y. However, it is important to understand the architecture of both neural networks and RNNs to be able to develop the LSTM model.

- **Neural Network:** A neural network is a collection of algorithms that are designed to identify patterns and trends. They are made up of a massive number of densely linked neurons that come together to resolve an issue. A neuron is a combination of a linear classifier and an activation function. The network is divided into three layers:

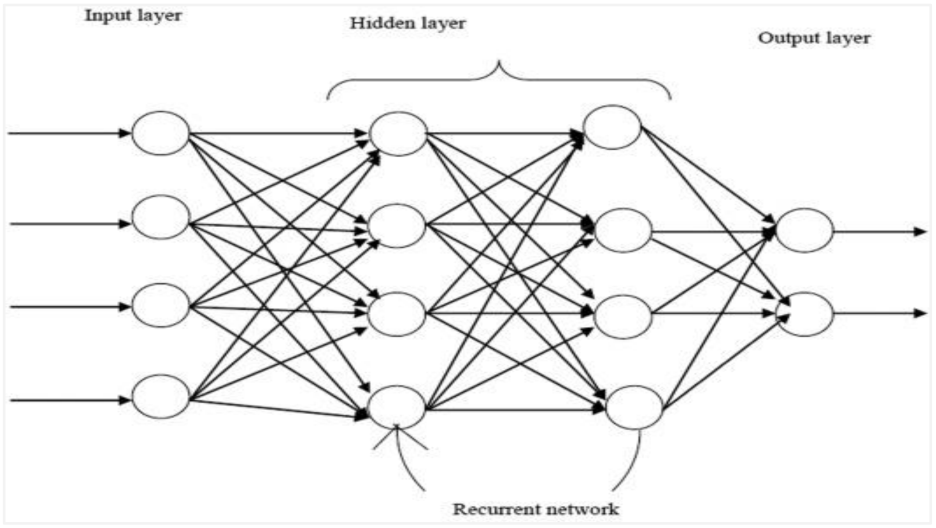
- ○ **Input layer:** this layer contains the data we feed into the model from external sources.
- ○ **Hidden layers:** there are multiple hidden layers which do all the computations and extracts all the features from the data.
- ○ **Output layer:** this layer takes input from preceding layers and comes to a final prediction based on the model's learning.

Figure 1: *Neural Network Architecture*



- **Recurrent Neural Network:** RRN is a feedforward neural network modification with internal memory. It is recurrent in structure because it executes the same function for each input, and the outcome of the current input is dependent on the previous calculation. After the output is generated, it is replicated and transmitted back into the recurrent neural network. It examines the current input and the output that has trained from the previous time step when making a decision.

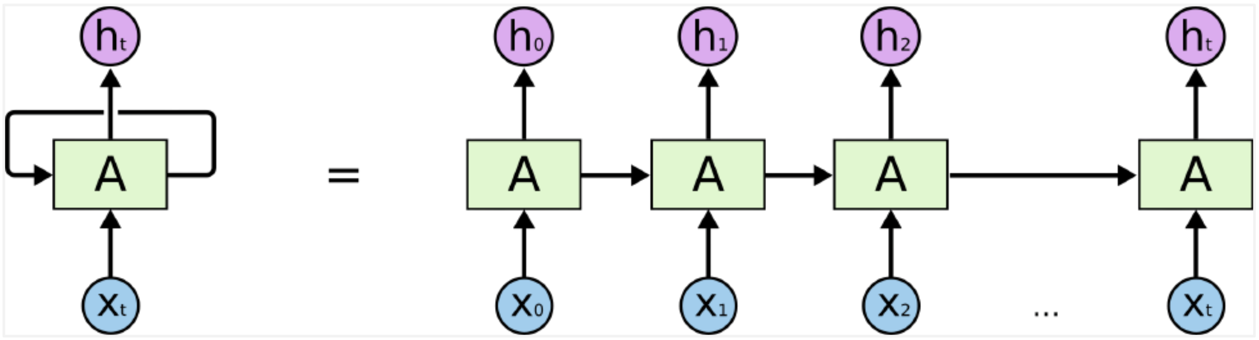
Figure 2: Recurrent *Neural Network Architecture*



4.1 Baseline: Unidirectional LSTM

A LSTM network is an enhanced RNN that includes a memory cell or cell state that is passed down the timesteps. At each timestep, the unit determines to discard some information from the cell and replace it with new information from the current input. This makes it simpler to recall previous data in memory, enhancing the model's performance by allowing it to recover useful information over long distances.

Figure 2: *Unidirectional LSTM Architecture*



To build our model, we have implemented the following steps:

- **Sequence the text**: The text will be sequenced by taking a series of phrases and assigning number tokens to them depending on the training set.

- **Word Embedding:** Word embedding is a technique for representing words for textual analysis on the basis of real-valued vectors that computers use to understand. We believe that word embeddings are the ideal way to represent text as they minimize the complexity of matrices and take into account semantic information, resulting in improved predictions.

- **Unidirectional Encoder and Decoder:** An LSTM cell serves as the encoder. It is given an input sequence over time and attempts to capture all of its information and store it in its ultimate internal state: hidden state and cell state. The internal states are then transmitted to the decoder, which attempts to generate the expected summary.

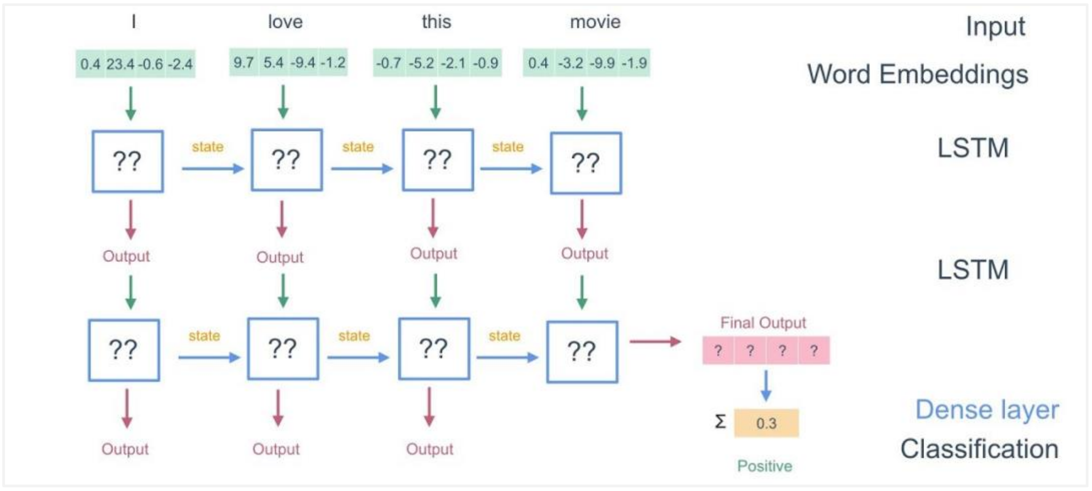
- **Inference Model:** The inference likewise incorporates an encoder and a decoder. It encodes the full input vector and configures the decoder with encoder internal states, which produces predictions by utilizing part of the learned model's layers.

We have utilized the following libraries to accomplish the above mentioned tasks: tensorflow, pandas, matplotlib, numpy, nltk, pickle, sklearn and keras.

4.2 Model: Hybrid LSTM

The Hybrid LSTM adopts the same methodology as the unidirectional LSTM. However, for both the EncoderDecoder and the inference model, it mixes a bidirectional encoder LSTM with a unidirectional decoder LSTM. We anticipate that the hybrid LSTM will outperform because, through employing a bidirectional encoder, it will read the entire phrase and thereafter proceed to map vectors to words. The architecture of a hybrid LSTM model is depicted in the diagram below:

Figure 2: *Hybrid LSTM Architecture*



4.3 Feature Engineering

Feature engineering is one of the key steps for building an effective abstractive summarization model. We have mainly used Vectorization which is a technique that is used for converting input data from its raw format into vectors of real numbers which can be understood by the model. This exposes the structure of the concept to the algorithm and will work well with the structure of the model the algorithm will create. In addition, we have also included simple meta features such as length of the text, as well as the length of the summary which are demonstrated in the graph below. This enabled us to effectively pad the sequence by adding zeros post the sequence. It is critical to make vectors of the same length for building the word embeddings and to ensure that the model will function.

5 Evaluation and Results

We employed Recall-Oriented Understudy for Gisting Evaluation (ROUGE), which is made up of a collection of criteria for assessing automatic text summarization and machine translation. ROUGE analyzes a model by comparing an automatically generated summary or translation to a set of human-generated reference summaries. The results of both models are shown below:

Table 1: *ROUGE Scores For Unidirectional and Hybrid LSTM*

	Metric	ROUGE Score
Unidirectional LSTM	ROUGE-L Average_R	0.22931
	ROUGE-L Average_P	0.25830
	ROUGE-L Average_F	0.29142
Hybrid LSTM	ROUGE-L Average_R	0.2531
	ROUGE-L Average_P	0.23273
	ROUGE-L Average_F	0.34172

As seen in the table above, both of our models performed poorly when compared to other LSTM models, which scored between 36% and 45% on all three measures. We believe this is due to the insufficient dataset utilized to train our model. We believe that this has a significant impact on our outcomes since neural networks typically require a large amount of training data due to the large number of parameters that must be tweaked by the learning process. This involves retaining the forward passes from activation functions so that the error gradients in the backward pass may be calculated. Moreover, we believe that the quality of the summaries generated are not representative of the concept of the text.

In addition, we can also see that the hybrid LSTM model performed slightly better than the unidirectional model as expected. This is because it will read the entire phrase and thereafter proceed to map vectors to words using the bidirectional encoder. However, the difference between these models isn't significant, which can also be explained by the limited dataset that we had to train our model.

6 Conclusion

Abstractive text summarization is still an area of improvement due its complexities. This is because abstractive summarization involves reducing the length of the text and can be expressed in many different ways while preserving the meaning of the text. According to our results, we believe that our abstractive summarization model is not effective in producing high quality summaries that best represent the full text. However, further research could investigate these models with much larger dataset, like CNN/Dailynews dataset which has been widely used to train abstractive summarization models. This is to better understand the effectiveness of the model to generate useful summaries. In addition, we recommend to also build a bidirectional LSTM, as well as add richer features of the model such as convolutional neural network.

References:

Cai, T., Shen, M., Peng, H., Jiang, L., & Dai, Q. (2019). Improving transformer with sequential context representations for abstractive text summarization. *Natural Language Processing and Chinese Computing*, 512–524. https://doi.org/10.1007/978-3-030-32233-5_40

Chen, Y.-C., & Bansal, M. (2018). Fast abstractive summarization with reinforce-selected sentence rewriting. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. https://doi.org/10.18653/v1/p18-1063

Ek, A., Bernardy, J. P., & Chatzikyriakidis, S. (2020, June). How does punctuation affect neural models in natural language inference. In *Proceedings of the Probability and Meaning Conference (PaM 2020)* (pp. 109-116)

Gupta, S., & Gupta, S. K. (2019). Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, *121*, 49–65. https://doi.org/10.1016/j.eswa.2018.12.011

J. H. Bappy, C. Simons, L. Nataraj, B. S. Manjunath and A. K. Roy-Chowdhury, "Hybrid LSTM and Encoder–Decoder Architecture for Detection of Image Forgeries," in IEEE Transactions on Image Processing, vol. 28, no. 7, pp. 3286-3300, July 2019, doi: 10.1109/TIP.2019.2895466.

Lab7. (n.d.). *The Essential Guide to Neural Network Architectures*. V7. Retrieved November 10, 2022, from https://www.v7labs.com/blog/neural-network-architectures-guide

Nallapati, R., Zhou, B., dos Santos, C., Gulcehre, C., & Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence RNNS and beyond. *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. https://doi.org/10.18653/v1/k16-1028

Song, S., Huang, H., & Ruan, T. (2018). Abstractive text summarization using LSTM-CNN based Deep Learning. *Multimedia Tools and Applications*, *78*(1), 857–875. https://doi.org/10.1007/s11042-018-5749-3