Tribhuvan University
Institute of Science and Technology

"Abstractive Text Summarization Using Recurrent Neural Network With Attention Based Mechanism"

Dissertation

Submitted To:

Central Department of Computer Science and Information Technology
Tribhuvan University, Kirtipur

Kathmandu, Nepal

In partial fulfillment of the requirements

for the Master's Degree in Computer Science and Information Technology

Submitted By:
Rojina Maharjan
13th July,2020

Tribhuvan University
Institute of Science and Technology

"Abstractive Text Summarization Using Recurrent Neural Network With Attention Based Mechanism"

Dissertation

Submitted To:

Central Department of Computer Science and Information Technology
Tribhuvan University, Kirtipur

Kathmandu, Nepal

In partial fulfillment of the requirements

for the Master's Degree in Computer Science and Information Technology

Submitted By:
Rojina Maharjan
13th July,2020

Supervisor:
Mr. Tej Bahadur Shahi

Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science and Information Technology

Student's Declaration

I hereby declare that I am the only author of this work and that no sources other than the listed here have been used in this work.

… … … … … … … …

Rojina Maharjan

13th July,2020

Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science and Information Technology

Supervisor's Recommendation

I hereby recommend that this dissertation prepared under my supervision by **Ms. Rojina Maharjan** entitled **"Abstractive Text Summarization Using Recurrent Neural Network With Attention Based Mechanism"** in partial fulfillment of the requirements for the degree of M.Sc. in Computer Science and Information Technology be processed for the evaluation.

… … … … … … … … … … … …

Mr. Tej Bahadur Shahi

Asst. Prof., CDCSIT

Tribhuvan University, Nepal

Date: 13th July, 2020

Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science and Information Technology

Letter of Approval

We certify that we have read this dissertation and in our opinion it is satisfactory in the scope and quality as a dissertation in the partial fulfillment for the requirement of Masters Degree in Computer Science and Information Technology.

Evaluation Committee

… … … … … … … … … … … …

Asst. Prof. Nawaraj Paudel

Head of Department

Central Department of Computer Science

And Information Technology

Tribhuvan University

Kirtipur, Nepal

… … … … … … … … … … … …

Mr. Tej Bahadur Shahi

Asst. Prof.

Central Department of Computer Science

And Information Technology

Tribhuvan University

Kirtipur, Nepal

… … … … … … … … … … … …

(External Examiner)

… … … … … … … … … … … …

(Internal Examiner)

Date:13th July,2020

ACKNOWLEDGEMENT

I would like to express my sincere thanks to my supervisor **Mr. Tej Bahadur Shahi**, Asst.Prof., CDCSIT for his continuous support, motivation, suggestions and guidance. His advice was inevitable and with his help I was able to work on my own field of interest and complete my dissertation.

I am also thankful to **Mr. Nawaraj Paudel**, Head of Department, CDCSIT who has provided all the help and facilities, which I required, for the completion of my dissertation.

Moreover, I would like to express my heartfelt gratitude to all my teachers at Central Department of Computer Science and Information Technology, Tribhuvan University who have imparted knowledge in various subjects. I want to express my gratitude to everyone who supported me throughout the course of M.Sc.CSIT.

Finally, I would like to express my thanks to my family and my friends for their direct and indirect support and encouragement for the completion of this dissertation.

Rojina Maharjan

13th July,2020

ABSTRACT

To facilitate the task of reading and searching information, it became necessary to find a way to reduce the size of documents without affecting the content of original document. The solution to this problem is abstractive text summarization system. Abstractive means that they are not restricted to simply selecting and rearranging passages from the original text. It is an automated system which takes an input text to produce another shorter version of given source documents without losing relevant data and meaning conveyed by the original text maintaining its semantic and grammatical correctness. This research aims to propose the RNN Encoder Decoder model with LSTM using attention mechanism for abstractive summarization to generate the summary carrying meaningful representation of the source document. ROUGE metric has been calculated to evaluate the summaries generated by the model run in 64 batch size with 256 hidden unit forming one layer.ROUGE1 and ROUGE2 score has been calculated among different metrics. The average ROUGE1 score obtained for validation data is 17.60 and ROUGE2 is 1.65.

Keywords: Recurrent Neural Network (RNN), Long Short Term Memory (LSTM),ROUGE (Recall-Oriented Understudy for Gisting Evaluation)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBERVATIONS

NN – Neural Network

RNN – Recurrent Neural Network

LSTM – Long Short Term Memory

NLP – Natural Language Processing

BRNN– Bidirectional Recurrent Neural Network

GPU – Graphics Processor Unit

ROUGE–Recall Oriented Understudy for Gisting Evaluation

EC2 –Elastic Compute Cloud

OOV –Out of Vocabulary

AWS – Amazon Web Services

GloVe – Global vectors for word representation.

CHAPTER 1
INTRODUCTION

1.1. Introduction

With the advent of the Internet, and the multiplicity of media mass storage, the amount of electronic documents and textual data became huge. The human, unable to manually handle large text volumes, must find automatic analysis methods adapted to automatic processing of personal data. These methods fall into the field of natural language processing (NLP). Among the most popular applications include machine translation, automatic summarizer, information retrieval, text mining, spell correction, speech synthesis, speech recognition or handwriting recognition, here we focus on automatic text summarization which allows user to decide whether the document is of interest or not, without looking at the whole document by extracting brief information without losing the meaning and important information in the original text.

Summarization is the task of condensing a piece of text to a shorter version that contains the main information from the original. There are two broad approaches to summarization: extractive and abstractive. Extractive methods assemble summaries exclusively from passages (usually whole sentences) taken directly from the source text, while abstractive methods may generate novel words and phrases not featured in the source text as a human written abstract usually does. The extractive approach is easier, because copying large chunks of text from the source document ensures baseline levels of grammaticality and accuracy.

Abstractive text summarization is the task of generating a headline or a short summary consisting of a few sentences that captures the salient ideas of an article or a passage. We use the adjective ‘abstractive’ to denote a summary that is not a mere selection of a few existing passages or sentences extracted from the source, but a compressed paraphrasing of the main contents of the document, potentially using vocabulary unseen in the source document. Abstractive summarization is closer as what we human usually do. We know that we human first conceive the text then compare it with our memory and related information, and then re-create its core in a brief text. That is why the abstractive summarization is more challenging than the extractive method, as the model should break the source corpus apart to the very tokens and regenerate the target sentences. Achieving meaningful and grammatically correct sentences in the summaries is a big deal that demands highly precise and sophisticated models.

1.2. Problem Statement

The sophisticated abilities that are crucial to high-quality summarization, such as paraphrasing, generalization, or the incorporation of real-world knowledge, are possible only in an abstractive framework. The key challenge in abstractive summarization is to optimally compress the original document in a lossy manner such that the key concepts in the original document are preserved. The summary is typically very short and does not depend very much on the length of the source document in this abstractive summarization. The main problem could be the summaries which are often repetitive and absurd.

1.3. Objective

The objective of this dissertation is:

- To generate summaries of the given CNN dataset applying encoder-decoder RNN with LSTM using attention and analyzing the summaries.
- To evaluate the generated summary using ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric evaluation.

1.4. Structure of Report

This report is organized in six chapters which includes following chapters:

Chapter 1 of this dissertation work consists of introduction, problem statement, objectives and limitations. Introduction focuses on the overview of the summarization and its types and describes why summarization is needed. Problem statement gives the analysis of the generation of repetitive and absurd summaries. The main objective and limitations of the abstractive summarization is also included in this chapter.

Chapter 2 describes the background study for the research and explanation of previous studies in this related work under literature review.

Chapter 3 includes the methodology of implementing abstractive text summarization using seq2seq RNN model with attention based mechanism, analysis criteria and all the details of the dataset.

Chapter 4 contains all the implementation details.
Chapter 5 contains analysis of the result.
Chapter 6 provides final conclusion and future works of the study.

CHAPTER 2

BACKGROUND STUDY AND LITERATURE REVIEW

2.1. Background Study

2.1.1. Deep Learning Approach

Over the past few years, deep neural networks have become extremely popular. Deep learning is the emerging field of computer science which is based around the concept of biological neural networks. It is all about using neural networks with more neurons, layers, and interconnectivity. Deep learning scientists and engineers try to mathematically describe various patterns from biological nervous systems. It is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence [7]. It is a subset of machine learning that enables computers to solve more complex problems [7]. We are using deep learning based models that map an input sequence into another output sequence, called sequence-to-sequence models using attentional recurrent neural network (RNN) encoder decoder LSTM model here for abstractive summarization. These models have been successful in many problems such as machine translation, speech recognition, video captioning and more.

2.1.2. Recurrent Neural Network (RNN)

RNNs use the idea of processing sequential information. The term “recurrent” applies because they perform the same task for every element of a sequence, with the output being depended on the previous computations. We assume that all inputs and outputs are independent of each other in a traditional neural network which could be bad idea for many tasks like if we want to predict the next word in a sentence we better know which words came before it. Recurrent Neural Network address this issue. Generally, a fixed-size vector is produced to represent a sequence by feeding tokens one by one to a recurrent unit. Another way to think about RNNs is that they have a “memory” which captures information about what has been calculated so far. This template is naturally suited for many NLP tasks such as language modeling, machine translation, speech recognition, image captioning. Here it has been used for abstractive text summarization. Here is what a typical RNN looks like:

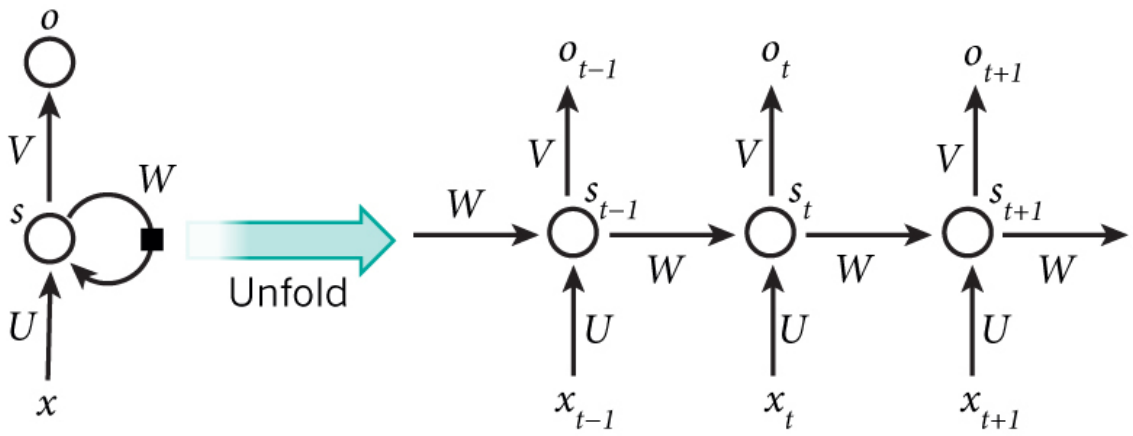


Figure 2.1: A recurrent neural network and the unfolding in time of the computation involved in its forward computation [6].

The above diagram shows a RNN being unrolled (or unfolded) into a full network.

The formulas that govern the computation happening in a RNN are as follows:

Here,

- x_t is the input at time step t,
- s_t is the hidden state at time step t .It's the memory of the network and s_t is calculated based on the previous hidden state and the input at the current step :
    - $s_t = f(Ux_t + Ws_{t-1})$
- The function f usually is a nonlinearity such as tanh or Rectified Linear Unit (ReLU).
- s_{t-1} is required to calculate the first hidden state, is typically initialized to all zeroes.
- o_t is the output at time step t.

2.1.3. Bidirectional Recurrent Neural Networks

An RNN processes the data in timesteps which indicates that the sequence is series of a continuous sampling of a variable and that sequence might be completely time-independent. The elements of a series are scanned to find their relation by RNN and this scan has only backward direction. RNNs have a memory of the past so they look at the previous timesteps to find the bounds and relations. But if an element in a sequence is not only related to the past, but to the next step, or to predict the future step, a Bidirectional Recurrent Neural Network (BRNN) is used. It analyze the dependencies of each element in a sequence to the previous and the next one.

2.1.4. Long Short-term Memory

RNNs are not able to learn unbalanced pairs of sequences in which the target length is different from the source. In each time-step an RNN loops back the state of its hidden layer. So in the next step, it has a memory of the past step. We can imagine a sequence of length 10 as our input and in the 2nd step the RNN can remember the state 1. But as it progresses in the length of the sequence, each time a new input combines with the looped back state, so in the step 10 the network has a memory of all the previous steps with the notion that they do not have the same strength. The first step can almost dissolve after this long repetitive incoming of new inputs. Therefore an RNN can remember the immediate past better, or with a better interpretation, it has a short-term memory [18]. Equipping these RNN architectures to long-term memory was the key idea of LSTM cells. Long Short Term Memory networks usually just called "LSTMs" are a special kind of RNN, designed to avoid the long-term dependencies. They were introduced by Hochreiter & Schmidhuber in 1997, and were refined and popularized by many people in following work.

LSTM cells are rather complex structures based on RNNs. We just explain two main parts of them that are two handles for keeping or forgetting the memory. A line that all the states can freely pass across it till the last timestep, and a forget gate. The forget gate itself is controlled by a 1 simple feed forward network, usually one layer that is fed with the previous hidden state. By training these gates, the LSTM will learn when to keep, and when to forget. It all depends on the pattern of the input data. For any input pattern if it is more frequent in the dataset, network will learn to keep its memory. LSTM, unlike the RNN, does it regardless of the length of pattern, long or short term memories will retain if they are more frequent. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

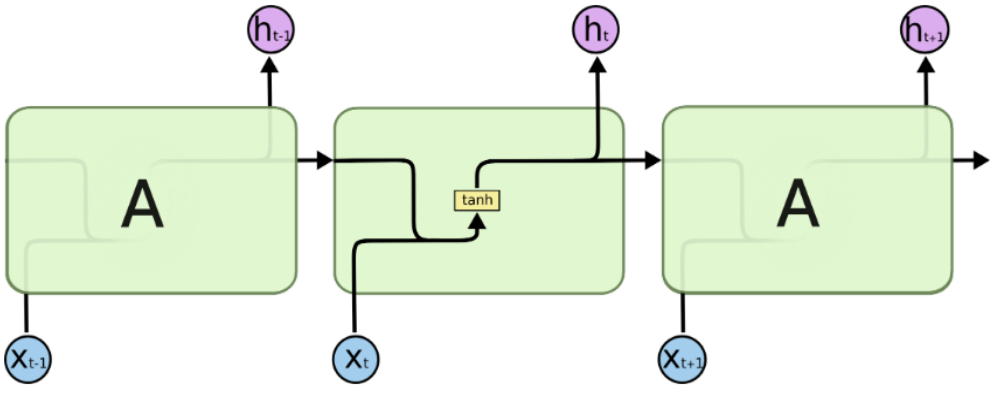


Figure 2.2: The repeating module in a standard RNN contains a single layer[2].

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.

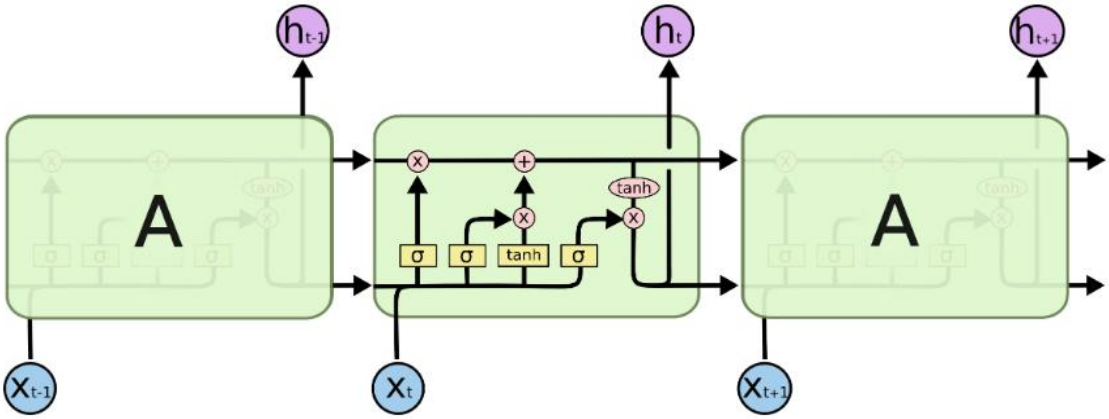
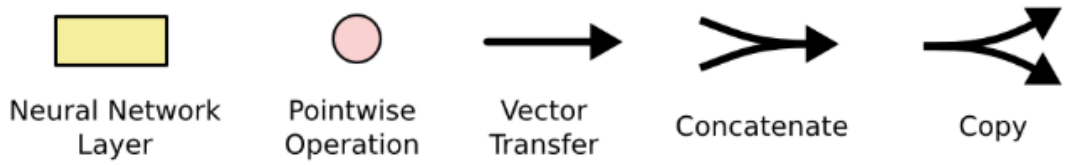


Figure 2.3: The repeating module in an LSTM contains four interacting layers[2].

The notations used in above figure are shown below:



In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations [2]. There are three gates in an LSTM.

Forget Gate Layer

$$f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right)$$

Input Gate Layer

$$i_t = \sigma\left(W_t\left[h_{t-1}, x_t\right] + b_i\right)$$

Vector of new candidate values

$$\widetilde{C}_t = \tanh\left(W_i\left[h_{t-1}, x_t\right] + b_c\right)$$

$$C_t = \widetilde{C}_t \otimes t_t + C_t$$

Output Gate Layer

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

New Hidden State

$$h_t = o_t * \tanh\left(C_t\right)$$

2.1.5. Word Embedding Methods

Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. Word embedding method learn a real-valued vector representation for a predefined fixed sized vocabulary from a corpus of text. It give us a way to use an efficient, dense representation in which similar words have a similar encoding. There are different techniques of word embeddings like Word2Vec and Glove. Here, we are using GloVe Embedding which is an unsupervised learning algorithm developed by Standford for obtaining vector representation for words. GloVe stands for global vectors for word representation.

2.2. Literature Review

Chen et al look into several approaches in order to improve an already existing sequence-to-sequence model. The state-of-the-art models rely on using the first two sentences of articles in order to provide the best possible summarization, they use LexRank in order to more generalize this approach and guarantee that they are always using the most relevant sentences in their input. Next, they use an additional attention model, CopyNet, in order to be able to use the source text more effectively and avoid the over usage of <UNK> and to copy salient sections of the input text directly into the summarization, such as names, dates, and rare words. CNN/DailyMail dataset is chosen for summarization task [4].

The model proposed by Chopra et al. could be seen as an extension of the proposed model for the same problem by Rush et al.. While Rush et al. use a feed-forward neural language model for generation, they use a recurrent neural network. Furthermore, encoder is more sophisticated, in that it explicitly encodes the position information of the input words. Lastly, the encoder uses a convolutional network to encode input words. These extensions result in improved performance [5].

Khatri et al. proposed a novel Document-Context based Seq2Seq models using RNNs for abstractive and extractive summarizations. This is similar to humans reading the title, abstract or any other contextual information before reading the document. This gives humans a high-level idea of what the document is about. They use this idea and propose that Seq2Seq models should be started with contextual information at the first time-step of the input to obtain better summaries. In this manner, the output summaries are more document centric, than being generic, overcoming one of the major hurdles of using generative models [9].

Kim and Huang explore the abstractive text summarization using attentive LSTM encoder-decoder recurrent neural network to obtain sensible result using the news articles form English Gigaword as dataset with some potential tweaks to the conventional architecture and comparing the results[10].

Nallapati et al., model abstractive text summarization using Attentional Encoder-Decoder Recurrent Neural Networks and show that they achieve state-of-the-art performance on two different corpora. They propose several novel models that address critical problems in summarization that are not adequately modeled by the basic architecture, such as modeling key-words, capturing the hierarchy of sentence to word structure, and emitting words that are rare or unseen at training time[12].
Paulus et al introduce a neural network model with a novel intra-attention that attends over the input and continuously generated output separately and a new training method that combines standard supervised word prediction and reinforcement learning (RL).They evaluate this model on CNN/Daily Mail dataset and NewYork Times dataset. Li et al attempts to replicate the same neural network model proposed by Paulus et al [14].

The first to apply modern neural networks to abstractive text summarization, achieving state-of-the-art performance on DUC-2004 and Gigaword, two sentence-level summarization datasets were Rush et al.. They explore a fully data-driven approach for generating abstractive summaries, combine a neural language model with a contextual input encoder inspired by the success of neural machine translation. Encoder is modeled off of the attention-based encoder of Bahdanau et al. in that it learns a latent soft alignment over the input text to help inform the summary. Crucially both the encoder and the generation model are trained jointly on the sentence summarization task [16].

Sabahi et al are the ones to use bidirectional encoder-decoder topology and bidirectional beam search for the summarization task. Their model differs from the previous models in three important ways: first, both the encoder and the decoder are bidirectional LSTMs. Second, they found that it extremely valuable to reverse the order of the input sequence. Third, a bidirectional approximate inference algorithm (BBS) is used to generate the final output, the summary [17].

See et al. proposed a novel architecture that augments the standard sequence-to-sequence attentional model in two orthogonal ways. First, they use a hybrid pointer-generator network that can copy words from the source text via pointing, which aids accurate reproduction of information, while retaining the ability to produce novel words through the generator. Second, they use coverage to keep track of what has been summarized, which discourages repetition. They apply the model to the CNN / Daily Mail summarization task, outperforming the current abstractive state-of-the-art by at least 2 ROUGE points [21].

Feng et al. proposed an attentive encoder-based extractive text summarization (AES) model to generate article summaries. AES can generate a rich document representation by considering both the global information of a document and the relationships of sentences in the document. A unidirectional recurrent neural network (RNN) and a bidirectional RNN are considered to construct the encoders, giving rise to unidirectional attentive encoder-based summarization (Uni-AES) and bidirectional attentive encoder-based summarization (Bi-AES), respectively. This experimental results show that Bi-AES outperforms Uni-AES [8]. Sinha et al. propose a fully data-driven approach using feed-forward neural networks for single document extractive summarization. The proposed model is scalable and is able to produce the summary of arbitrarily sized documents by breaking the original document into fixed sized parts and then feeding it recursively to the network [22].

Yu et al. focused on approaches to building a text automatic summarization model for news articles, generating a one-sentence summarization that mimics the style of a news title given some paragraphs. They managed to build and train two relatively complex deep learning models that outperformed our baseline model, which is a simple feed forward neural network exploring Recurrent Neural Network models with encoder-decoder using LSTM and GRU cells, and with/without attention[25].

CHAPTER 3
RESEARCH METHODOLOGY

3.1. Data Collection

Varieties of popular datasets have been used in the past and one famous dataset to use is CNN Dataset. Here, we are using CNN dataset which is comprised of more than 92 thousand text documents with CNN stories .Each article comes with a short set of summarized bullet points that represent meaningful “highlights” of the piece. As a result, the accuracy of generated summaries is checked with the given highlights of the dataset. The CNN dataset was downloaded from New York University, in the version made available by Kyunghyun Cho.

3.1.1. Preprocessing

The raw text need to be preprocessed before feeding it into the neural networks so at first the text are splitted into an article and a summary part. Each of the model will be trained using the CNN data which needs to be preprocessed like removing punctuations, removing multiple arranged white spaces, removing start and end phrase of each article, e.g. location, author, copyright, removing ’@highlight’ for the summary part, replacing English contractions by the full version of the corresponding expression like ’aren’t’ with ’are not’, placing spaces before punctuation to separate them from words, removing stopwords, converting to lower case and create word vectors for each paragraph and its summary. The preprocessing helps in ensuring that the source text can be more easily converted into tokens, which are numerical representations of each single word or punctuation from the original source text. Each word is searched in a predefined dictionary which was created during the training phase.

3.2. System Architecture

The system architecture shows the process of implementing abstractive text summarization using Seq2Seq RNN model with attention-based mechanism. The output summary will be generated using Long Short Term Memory (LSTM) model. The model is based on a sequence-to-sequence Recurrent Neural Network architecture, including Luong Attention Mechanism. In this research, each RNN step is comprised by 256 LSTM (Long Short Term Memory) cells, forming only one layer.

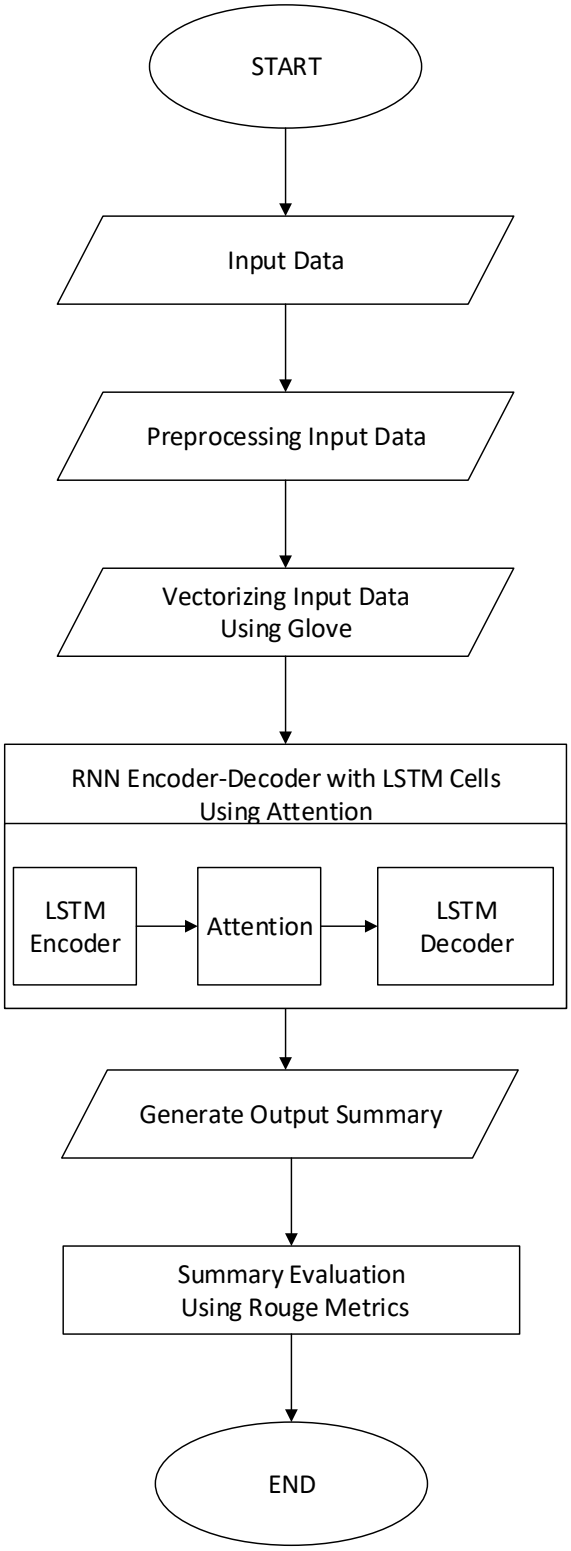


Figure 3.1: System Architecture for Abstractive Text Summarization

3.3. Seq2seq Model Architecture

A Seq2seq maps two sequences to each other that are not necessarily in the same size, in two steps: Compressing the first sequence, and then inferring the output from it. We are using this architecture which has two side named encoder and decoder that are both LSTM layers. The encoder-decoder LSTM architecture is comprised of two models:

Encoder: The encoder is responsible for reading the source document and encoding it to an internal representation.

Decoder: The decoder is a language model responsible for generating each word in the output summary using the encoded representation of the source document.

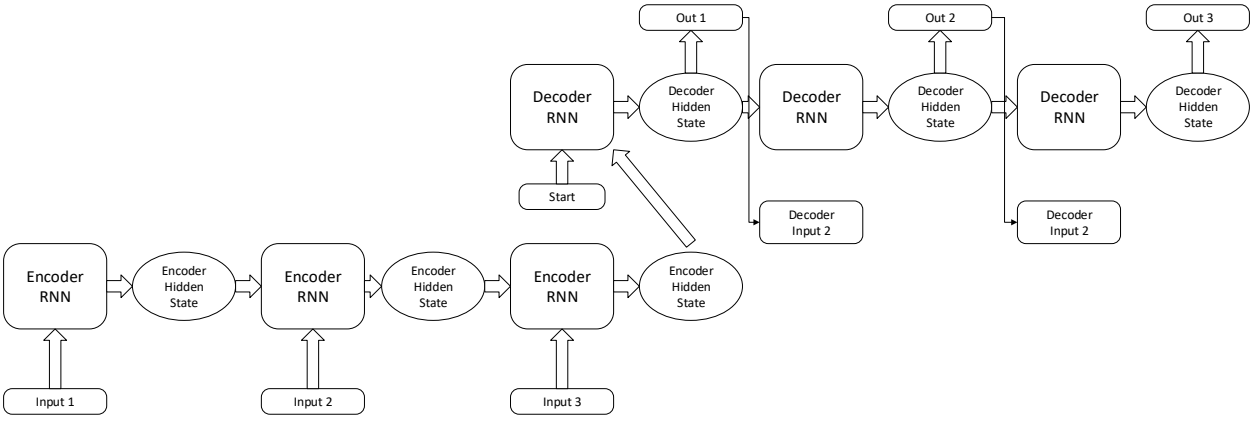


Figure 3.2: Overview of Sequence to Sequence Model

- An encoder is an RNN whose purpose is to turn an input sequence into a different representation that still contains all the information about the source which is then able to be used in different tasks. The encoder first works by taking a word and embedding it which allows us to capture the semantic meaning of a word and then turn it into a vector. This embedding will be the input into our RNN and is turned into a hidden state. Then this hidden state is used to create the next hidden state and so on until we get to the end of the source text. The encoded version of the input sequence is a representation of its information and its principal patterns. We assume it as a compressed memory of the whole elements of the input [4].

- The equation below explains the probability distribution of output sequence, given the input sequence. T and T' are the length of input sequences x and output sequence y respectively. s is the state of the last hidden layer in the encoder[18].

$$p\left(y_1, \ldots, y_{T'} | x_1, \ldots, x_T\right) = \prod_{t=1}^{T'} p\left(y_t | s, y_1, \ldots, y_{t-1}\right)$$

- Having the hidden state of the encoder in the first timestep of decoder, it gives us an output, which is a vector. We find the closest word vector in our dictionary to the output word vector. To measure how the output of that timestep matches the target word we can calculate a score from their multiplication. s_t and v_t are the hidden state of decoder and output layer's weight vector in timestep t [18].

$$e\left(w_t\right) = s_t v_t$$

- Now that we have the scores of outputs, we can calculate the probability of every output word. We use a normalization technique called softmax in the equation 3[18].

$$p\left(w_i | w_1, \ldots, w_{i-1}, s_t\right) = \frac{\exp\left(e\left(w_i\right)\right)}{\sum_j \exp(e(j))}$$

$e(w_i)$ is the score of the current output word.

- In each the model we select the most probable word in the output and put in the input of the next layer[18].

3.4. Improving Seq2seq Model with Attention Mechanism

A seq2seq model is combined with an encoder and a decoder. The decoder receives all the contents of the encoder compressed in one vector and trains the target with its presence. However, when the source sequence is too long, its content will fade in the coded vector and it is very hard to capture the essence of the entire input sequence in a single hidden state if the input sequence is longer .It makes sense to think that hypothetically the average vector of two big enough corpus are very similar to eachother. To solve this problem, attention mechanism

tries to get multiple encoded vectors from the source and not only one. The performance of sequential models improves to a higher extend if it is equipped with the attention mechanism.

At high level we can say, an attention mechanism in deep learning let our neural network just to focus on relevant parts of our input more than irrelevant parts. Attention is one component of a network's architecture, and is in charge of managing and quantifying the interdependence [31]:

- Between the input and output elements (General Attention)
- Within the input elements (Self Attention)

The type of attention we are using here for this summarization task is Luong Attention proposed by Thang Luong. It is referred to as Multiplicative Attention and was built on the top of Bahadanau attention which is referred to as Additive Attention.

The process of Luong attention mechanism is explained in step-by-step process as follows:

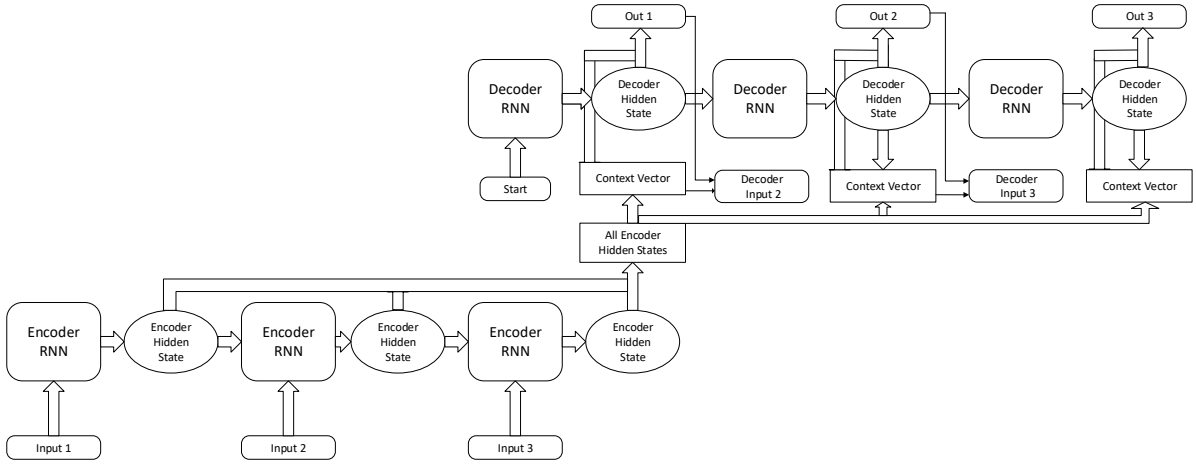


Figure 3.3: Overall process of Luong attention mechanism

3.4.1. Producing the Encoder Hidden States:

We are using LSTM, one of the variant of RNN to encode the input sequence. At first, input sequence is passed through the RNN encoder, a hidden state/output will be produced. An encoder is an RNN whose purpose is to turn an input sequence into a different representation that still contains all the information about the source which is then able to be used in different tasks. The encoder first works by taking a word and embedding it which allows us to capture the semantic meaning of a word and then turn it into a vector. This embedding will be the input into our RNN and is turned into a hidden state. Then this hidden state is used to create the next

hidden state and so on until we get to the end of the source text. The encoded version of the input sequence is a representation of its information and its principal patterns. We assume it as a compressed memory of the whole elements of the input.

3.4.2. Decoder RNN

The decoder in Luong Attention uses the RNN in the first step of the decoding process rather than the last. The RNN will take the hidden state produced in the previous time step and the word embedding of the final output from the previous time step to produce a new hidden state which will be used in the subsequent steps.

3.4.3. Calculating Alignment Scores

The alignment score is the essence of the Attention mechanism, as it quantifies the amount of "Attention" the decoder will place on each of the encoder outputs when producing the next output [31]. Alignment Scores are calculated using the new decoder hidden state and the encoder hidden state at each time step of the decoder.

In the encoder-decoder framework depicted in figure, given all the hidden states of the encoder i.e.

$$h^e = \left(h_1^e, h_2^e, \dots, h_j^e\right)$$

and the current decoder hidden state h_t^d,the alignment score $s\left(h_j^e, d_t^d\right)$ is obtained by the content-based score function[30].

There are three different ways of defining alignment scoring functions dot, general and concat and these scoring functions make use of encoder outputs and the decoder hidden state produced in the previous step to calculate the alignment scores.

- **Dot**

  The first one is the dot scoring function. To produce the alignment score, we need to take the hidden states of the encoder and multiply them by the hidden state of the decoder.

  $s\left(h_j^e, d_t^d\right) = (h_j^e)^T h_t^d$

- **General**

  The second type is called general and is similar to the dot function, except that a weight matrix is added into the equation as well.

  $s\left(h_j^e, d_t^d\right) = h_j^e W_{align} h_t^d$

- **Concat**

  The last function is slightly similar to the way that alignment scores are calculated in Bahdanau Attention, whereby the decoder hidden state is added to the encoder hidden states.

  $$s\left(h_j^e, d_t^d\right) = (v_{align})^T \tanh(W_{align}\left(h_j^e \oplus h_t^d\right) + b_{align})$$

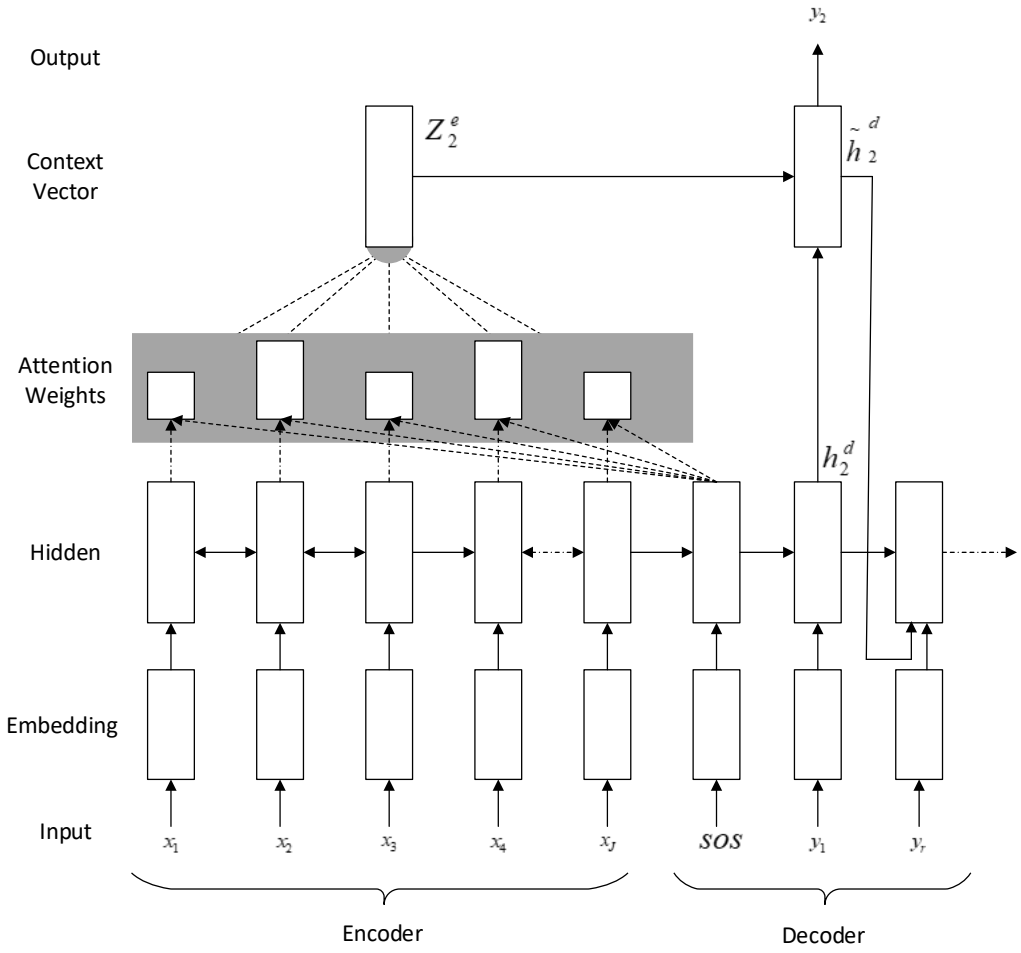


Figure 3.4: An attention based seq2seq model where each hidden unit represents an LSTM cell

3.4.4. Softmaxing the Alignment Scores

The alignment scores for each encoder hidden state are combined and represented in a single vector and then a softmax function is applied on this vector to obtain the attention weights. The softmax function will cause the values in the vector to sum up to 1 and each individual value will lie between 0 and 1, which will represent the weightage each input holds at that time step[31]. The attention distribution a_{tj}^e over the source tokens is calculated as follows:

$$a_{tj}^{e} = \frac{\exp(s_{tj}^{e})}{\sum_{k=1}^{j} \exp(s_{tk}^{e})}$$

3.4.5. Calculating the Context Vector

After computing the attention weights in the previous step, we can now generate the context vector by doing an element wise multiplication of the attention weights with the encoder outputs. Due to the softmax function in the previous step, if the score of a specific input element is closer to 1 its effect and influence on the decoder output is amplified, whereas if the score is close to 0, its influence is drowned out and nullified [31].With the attention distribution, we can naturally define the source side context vector for the target word as:

$$z_t^e = \sum_{j=1}^{J} \alpha_{tj}^e h_{tj}^e$$

3.4.6. Producing the Final Output

In the last step, the context vector obtained is concatenated with the decoder hidden state h_t^d we generated in step 2 and the attention hidden state is obtained as:

$$\tilde{h}_t^d = W_z\left(zt_t^e \oplus h_t^d\right) + b_z$$

3.5. Parameters Used

In this study, the model will be trained using following parameters.

- **Dropout:** Dropout refers to dropping out units (i.e. neurons) (both hidden and visible) during the training phase of certain set of neurons which is chosen at random. It is used to prevent the overfitting. In this study, 0.4 dropout is used.
- **Unit type:** The unit type refers to the type of hidden unit used in model. In this study, the hidden units will be Long Short Term Memory (LSTM) unit.
- **Encoder type:** Encoder type refers to the approach to be used to encode the input sentences. In this study, bidirectional encoder type has been used.
- **Decoder type:** Decoder used is unidirectional.

- **Learning Rate:** Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient. 0.001 learning rate is used in this study.
- **Optimizer:** The difference between the predicted output and the actual output is called as loss function or also as a cost function. Optimizers minimize the loss function updating the weight parameters. Among different optimizers, here Adam optimizer is used. Adam (Adaptive Moment Estimation) is computationally efficient and has very little memory requirement.

3.6. Summary Evaluation Metric

The performance of the model is evaluated with ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric which is a set of metrics for evaluating automatic summarization of texts as well as machine translation. It count the number of over-lapping units such as n-grams, word sequences, and word pairs between the machine-generated summary to be evaluated and the ideal summaries and loss graph. It works by comparing an automatically produced summary against a set of reference summaries [11]. ROUGE-N metric measures unigram, bigram, trigram and higher order n-gram overlap.

3.6.1. ROUGE-N

$$ROUGE - N = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count_{match}\ (gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

Where *n* stands for the length of the n-gram, *gramn*, and *Countmatch*(*gramn*) is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries[11].

It measures the overlap of N-grams between the system and reference summary. For the model, ROUGE1 and ROUGE2 metric are calculated and result is compared.ROUGE-1 refers to overlap of unigrams between the system summary and reference summary. ROUGE-2 refers to the overlap of bigrams between the system and reference summary. Here, perfect match represents results in a score of 100, whereas a perfect mismatch results in a score of 0 while it can be represented as range 0 to 1.

- **Recall**

Recall in the context of ROUGE means how much of the reference summary is the system summary capturing and it can be computed as:

$$Recall = \frac{number\_of\_overlapping\_words}{total\_words\_in\_reference\_summary}$$

- **Precision**

When a system summary is too long capturing all words in the reference summary but much of the words in the system summary is useless, precision is needed. Precision measures how much of the system summary is actually needed or fact relevant and it is computed as:

$$Precision = \frac{number\_of\_overlapping\_words}{total\_words\_in\_system\_summary}$$

- **F-Measure**

The F-measure or F-score that combines both measures Precision and Recall as the harmonic mean.

$$F - measure = \frac{2 \times precision \times recall}{precision + recall}$$

CHAPTER 4
IMPLEMENTATION

4.1. Implementation

The attention based recurrent neural network abstractive summarization system has been implemented in Python programming language with Tensorflow as inbuilt library.

- Python Programming

  Main coding and development has been done on python3. It is a general purpose, platform independent, multi-paradigm programming language which supports both Object-oriented programming and structured programming. The python code uses no compiler and can run on just about any device that runs the Python shell.

- Jupyter Notebook

  The Jupyter Notebook tool has been used to write python code which is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. It is used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. With jupyter notebook it is easier to share code as well as it documentation in single file.

- Tensorflow

  TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. TensorFlow 2.0 Beta version is used here.

- Test Environment

  For the testing, an Elastic Compute Cloud (EC2) server was launched on Amazon Web Services (AWS). Amazon EC2 is a web service that provides secure, resizable compute capacity in the cloud. It provides you with complete control of your computing

resources and lets you run on Amazon's proven computing environment. The server was Ubuntu 18.04 GPU instance with 8 virtual cores, 15 GB of RAM and 60 GB of SSD storage. The graphics used is NVIDIA 4 GB with dedicated Graphic Double Data Rate (GDDR). The jupyter notebook was launched inside the server and was accessed using the public IP of server. All the coding and development was done using the notebook.

4.2. Training the Model

In this research work, attention based recurrent neural network with Long Short Term Memory (LSTM) cells is used for abstractive text summarization. The encoder part of the RNN seq2seq model admit an upper limit of 400 tokens. The size is limited to 400 tokens and select approximately 20,000 articles short enough to fit the encoder. To produce a summary from a longer text, the model breaks input source text in pieces of maximum 400 tokens in length and then join the resulting summaries. For source text pieces whose length is less than 400 tokens, padding is done for the remaining token positions. Each RNN step is comprised by 256 LSTM (Long Short-Term Memory) cells, forming only one layer. The encoder is bi-directional and the decoder is unidirectional. The learning rate used for each experiment was 0.001 and the model is run in 64 batch size.

Finally, there is a post process step on the resulting summary to make it more readable. This involves capitalizing the first letter of words in the beginning of sentences and removing the spaces before punctuation. Missing words are represented by a special token "UNK".

4.3. Data Analysis

The total number of article unique tokens in the CNN dataset is 112683.The number of article and summary unique tokes is 114320. Pre-trained GloVe 100d is used as our word embedding vectors which are trained on aggregated global word-word co-occurrence statistics. The number of words missing from Glove is 1536 while the percentage of words that are missing from vocabulary defined is 1.34%.The total number of words in articles and summaries is 6941892 while the number of undefined tokens is 170965 i.e. 2.46%.For this model, we are using mini-dataset i.e 18000 articles of CNN dataset for training and for validation purpose we are using 2000 articles.

CHAPTER 5
RESULT AND ANALYSIS

5.1. Result

In this research work, Attention based Recurrent Neural Network with Long Short Term Memory (LSTM) cells is developed for abstractive text summarization. The purpose of this research is to generate the abstractive summary of the given article and to evaluate the generated summary using ROUGE metrics. Here, ROUGE-1, ROUGE-2 scores of generated summary is calculated for training and validation dataset.

5.2. Analysis

The performance of the model is evaluated with the calculation of ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric. For the model run for 82epochs, with hidden size 256 and learning rate 0.001, the maximum, minimum and average ROUGE-1, ROUGE-2 score among 200 generated summaries is given below:

Table 5.1: Maximum ROUGE-1 and ROUGE-2 Score (82 epochs)

CNN Dataset	ROUGE-1	ROUGE-2
Training Data	100	100
Validation Data	44.07	24.56

Table 5.2: Minimum ROUGE-1 and ROUGE-2 Score (82 epochs)

CNN Dataset	ROUGE-1	ROUGE-2
Training Data	3.33	0
Validation Data	0	0

Table 5.3: Average ROUGE-1 and ROUGE-2 Score (82 epochs)

CNN Dataset	ROUGE-1	ROUGE-2
Training Data	28.35	12.56
Validation Data	17.60	1.65

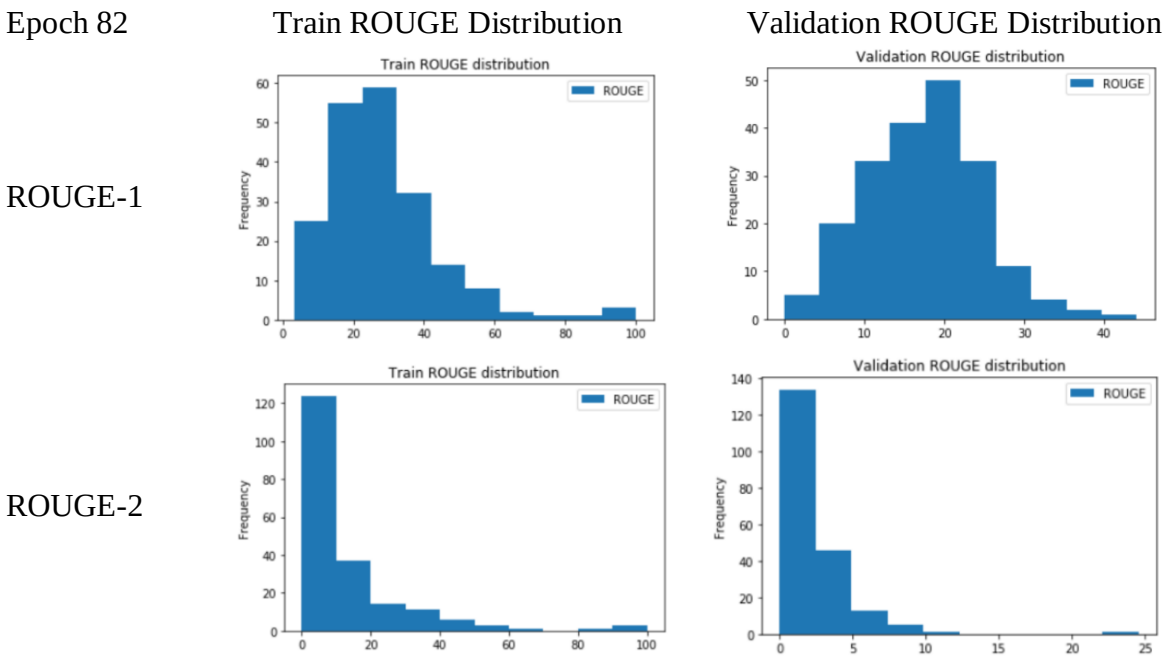


Figure 5.1: ROUGE-1 and ROUGE -2 Score distribution chart for generated summaries of training and validation dataset (82 epochs)

From the ROUGE distribution chart above we can see that the ROUGE-1 score of generated summaries for training data is spread to 100 but more of the summaries are below 60 score. While in validation dataset, score is low and there is higher number of summaries generated which have ROUGE-1 score below 30 even the highest is 44.07.Eventhough the highest ROUGE-2 score of the generated summaries for training data reach 100 but among 200 summaries generated the number of score is distributed below 40 and more summaries are generated which have score below 20 and for validation its even low than that. There are more number of summaries which have Rouge-2 score below 5.The average number of ROUGE score for validation is 17.65 and ROUGE-2 is 1.65.

For 96 epochs, the maximum, minimum and average ROUGE-1, ROUGE-2 score among 200 summaries generated is given below:

Table 5.4: Maximum ROUGE -1 and ROUGE -2 Score(96 epochs)

CNN Dataset	ROUGE-1	ROUGE-2
Training Data	100	100
Validation Data	44.12	21.21

Table 5.5: Minimum ROUGE -1 and ROUGE-2 Score (96 epochs)

CNN Dataset	ROUGE-1	ROUGE-2
Training Data	6.25	0
Validation Data	0	0

Table 5.6: Average ROUGE -1 and ROUGE-2 Score (96 epochs)

CNN Dataset	ROUGE-1	ROUGE-2
Training Data	29.15	13.62
Validation Data	16.66	1.47

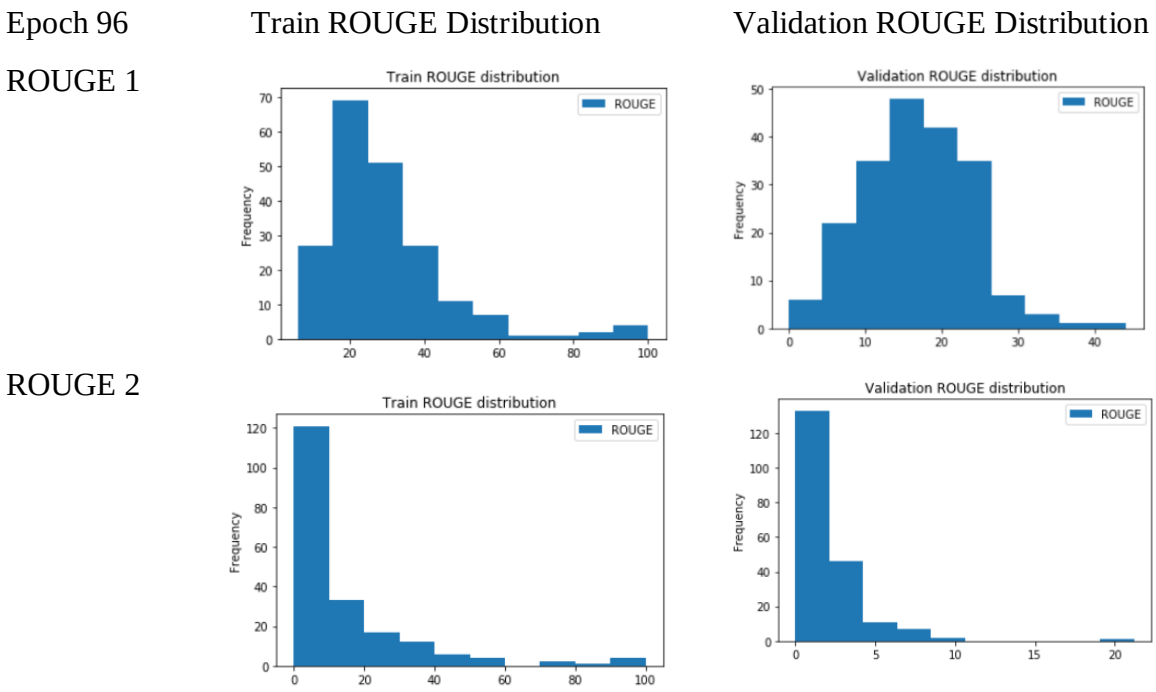


Figure 5.2: ROUGE -1 and ROUGE -2 Score distribution chart for generated summaries of training and validation (epochs 96)

From the ROUGE distribution chart above we can see that the ROUGE score of generated summaries for training data is spread to 100 but more of the summaries are below 60 score. While in validation dataset, score is low and there is higher number of summaries generated which have ROUGE-1 score below 30 with the highest being 44.12 and is more like a result obtained with the model run for 82epochs. ROUGE-2 scores are even lower and there are more number of summaries which have ROUGE-2 score below 5.There is small difference in the average scores generated with the model run for 96 epochs.

For 109 epochs, the maximum, minimum and the average ROUGE-1, ROUGE-2 score among 200 summaries generated is given below:

Table 5.7: Maximum ROUGE-1 and ROUGE-2 Score (109 epochs)

CNN Dataset	ROUGE-1	ROUGE-2
Training Data	100	100
Validation Data	37.04	15.09

Table 5.8: Minimum ROUGE-1 and ROUGE-2 Score (109 epochs)

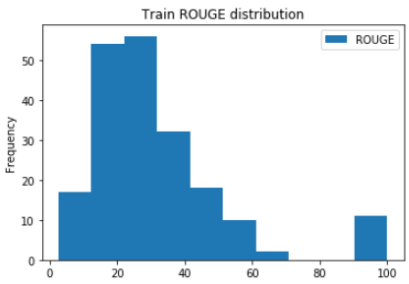
CNN Dataset	ROUGE-1	ROUGE-2
Training Data	2.63	0.00
Validation Data	0.00	0.00

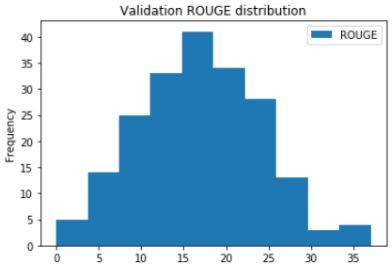
Table 5.9: Average ROUGE-1 and ROUGE-2 Score (109 epochs)

CNN Dataset	ROUGE-1	ROUGE-2
Training Data	31.69	17.49
Validation Data	16.84	1.38

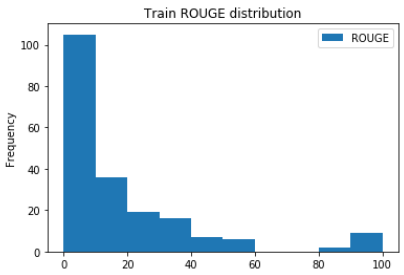
Epoch 109 Train ROUGE Distribution Validation ROUGE Distribution

ROUGE 1





ROUGE 2



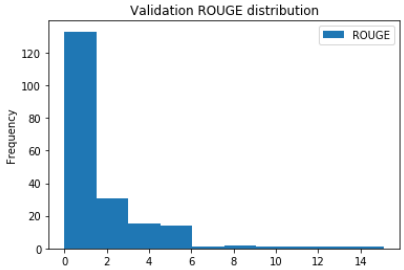


Figure 5.3: ROUGE -1 and ROUGE -2 Score distribution chart for generated summaries of training and validation dataset (epochs 109)

The average validation ROUGE score is lower in comparison to scores generated from the model run for 109 epochs. Analyzing all the average ROUGE scores generated, the training performed at higher number of epochs achieves lower ROUGE-2 score in relation to the scores generated at lower number of epochs i.e. 82 and 96 while ROUGE-1 average score is 16.84 which is slightly higher than the model run for 96epochs but still less than model run for 82 epochs. The model generated in the epoch 82 gives highest number of ROUGE-2 and highest ROUGE-1 score is obtained with the model generated in the epoch 96.

The snapshot of the summary generated with highest ROUGE-1, ROUGE-2 score i.e. 100 for training dataset is shown below:

```
SUMMARY with highest score (ROUGE = 100.00)

ARTICLE
after more than two decades together , kiss bassist gene simmons has married his
longtime girlfriend , according to his publicist . simmons , who is also the
star of the reality show gene simmons family jewels , has dated shannon tweed
for 28 years . the pair got hitched in beverly hills , said the publicist , dawn
miller . during his proposal in belize , featured in an earlier episode ,
simmons tells his love what she means to him . i come with so much baggage , but
you are the only friend i have got , you are the only one i have ever loved , he
said . i have never said those words to anybody . it is funny , i used to watch
movies where they say i cannot live without you , but for me it is true . the
proposal came after a nasty tiff between the two on the joy behar show this
summer . simmons joined in a joke about his philandering ways , ticking tweed
off . she stormed off the set . will gene and shannon work past their problems
and make it down the aisle ? the reality show website touts in a spotlight on
wedding episodes set to start tuesday . cnn breeanna hare and karen bonsignore
contributed to this report .

REFERENCE SUMMARY
simmons has dated shannon tweed for 28 years . the proposal comes after a nasty
tiff between the two this summer .

HYPOTHESIS SUMMARY
simmons has dated shannon tweed for 28 years . the proposal comes after a nasty
tiff between the two this summer .
```

Figure 5.4: Article, Reference Summary and Hypothesis Summary having highest ROUGE -1, ROUGE -2 score in the training dataset

Here, the ARTICLE is the CNN article whose summary is to be generated, REERENCE SUMMARY is the summary given with the dataset and HYPOTHESIS SUMMARY is the summary which we generate with this model. The highest ROUGE-1 and ROUGE-2 score for training data is 100 so the model produced exact same summary.

The snapshot of the generated summary with highest ROUGE-1 score i.e. 44.12 for validation data in the epoch 96 is shown below:

```
SUMMARY with highest score (ROUGE = 44.12)

ARTICLE
a 30-year-old uzbekistan national living in boise , idaho , was arrested
thursday on federal terrorism charges , the justice department said . a grand
jury in boise returned a three-court indictment charging fazliddin kurbanov , 30
, with one count of conspiracy to provide material support to a designated
foreign terrorist organization , one count of conspiracy to provide material
support to terrorists and one count of possessing an unregistered destructive
device . a grand jury in salt lake city , utah , returned a separate indictment
charging him with one count of distribution of information relating to
explosives , destructive devices and weapons of mass destruction . one of our
highest priorities is disrupting potential acts of terrorism . the coordinated
investigation , arrest , and indictments in this case demonstrate the commitment
of all involved to do just that , us attorney for the district of utah david b .
barlow said in a statement announcing the arrest . it was not immediately clear
whether kurbanov had retained counsel . according to the idaho indictment , he
is alleged to have knowingly conspired with others to support to the islamic
movement of uzbekistan , an american-designated foreign terrorist organization ,
between august and may . he is accused of having provided support , believing
that it was to be used in an offense involving the use of a weapon of mass
destruction , the statement said . in utah , he is alleged to have taught and
demonstrated how to make explosive devices , by showing internet videos and
conducting instructional shopping trips . kurbanov , who is in the united states
legally , is expected to make an initial appearance in court in boise on friday
.

REFERENCE SUMMARY
30-year-old fazliddin kurbanov is arrested in boise , idaho . he faces a three-
count indictment in idaho and a single-count accusation in utah . authorities
say he provided support to the islamic movement of uzbekistan .

HYPOTHESIS SUMMARY
30-year-old kurbanov pleads not guilty in boise , idaho . he faces a charge of
attempted assault in a murder charge , a source says . the cause of the death is
under investigation .
```

Figure 5.5: Article, Reference Summary and Hypothesis Summary having highest ROUGE -1 score in the validation dataset

This output summary has a high ROUGE-1 score due to correctly predicted words, however, the semantic meaning is quite different from the actual summary but the context looks similar. According to our sample output, the model is able to capture the key words in the input most of the times. However, the output meaning can be a bit off frequently due to inability to capture all of the semantic meaning of the main point of input paragraphs.

The snapshot of the generated summary with highest ROUGE-2 score for validating data in the epoch 82 is shown below:

```
SUMMARY with highest score (ROUGE = 24.56)

ARTICLE
an earthquake with a preliminary magnitude of 7.0 struck near mexico western
coast on wednesday , the us geological survey said . the epicenter of the quake
was 32 miles 51 kilometers southwest of la mira , michoacan , usgs said . the us
agency said the quake was about 40 miles 65 kilometers deep . mexico national
seismological service reported its depth at 10 miles 16 kilometers , and said
its magnitude was 64 residents felt the quake in mexico capital , 350 miles 560
km from the epicenter . there were no immediate reports of damages or injuries .
an aerial survey of the capital showed no major damage , mexico city mayor
marcelo ebrard said in a twitter post . cnn rey rodriguez and carloo mauricio
perez contributed to this report .

REFERENCE SUMMARY
there are no immediate reports of damages or injuries . residents feel the quake
in mexico city , hundreds of miles from the epicenter . aerial survey of the
capital shows no major damage , mayor says .

HYPOTHESIS SUMMARY
the usgs revises down the preliminary evacuation sets , official says . there
are no immediate reports of damage . the quake was centered near the eastern
town of .
```

Figure 5.6: Article, Reference Summary and Hypothesis Summary having highest ROUGE -2 score in the validation dataset

Eventhough for ROUGE-N, bigger N implies more fluency in the summary because matching with a bigger portion of the reference summary which is more fluent, implies more fluency, it is only applicable in extractive text summarization where summary is generated extracting important sentences from the input document but not in the abstractive text summarization where we aren't direct reusing the sentences. ROUGE scoring expects system summaries to be exactly identical to the reference summaries and identical summaries are rare in reality as there are different ways to express the same essential content and different set of connective words and intensifiers may be used to express the same thing and also it doesn't capture synonymous concepts. The output summary has highest ROUGE-2 score but it is just 24.56 because there will be few overlapping bigrams incase of abstractive summarization where we are not reusing the sentences. Even the score is low, the context of summary generated looks similar with the reference summary.

The learning curve shown below is generated when the model run for epochs 82 with batch size of 64.

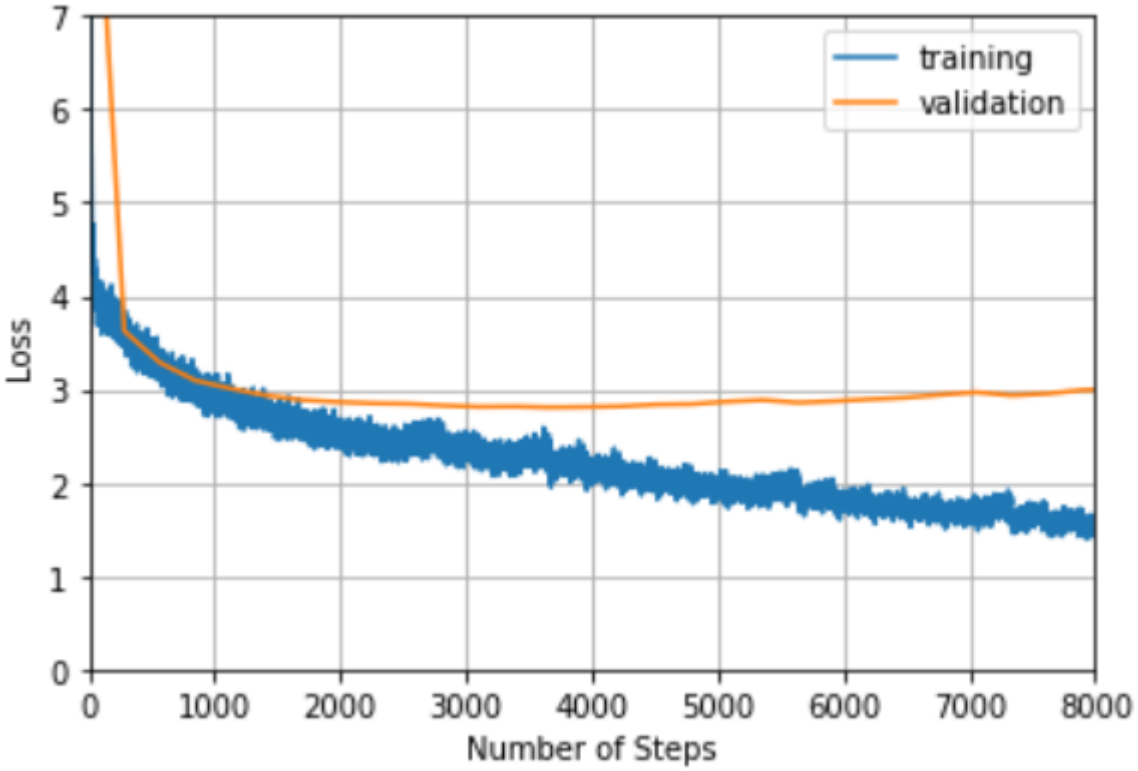


Figure 5.7: Learning Curve

CHAPTER 6
CONCLUSION

6.1. Conclusion

In this research, 200 summaries of the 20000 articles of CNN dataset are generated applying encoder-decoder RNN with LSTM using attention and these generated summaries are analyzed using ROUGE-N metrics. Among different ROUGE-N metrics, ROUGE1 and ROUGE2 score are calculated to evaluate the generated summary. The maximum, minimum and average ROUGE1 and ROUGE2 score is generated for the model run for different epochs. The average ROUGE1 score obtained with the model run for 82 epochs for validation data is 17.60 and ROUGE2 score is 1.65.The highest ROUGE1 score generated with the model run for 96 epochs is 44.12 and highest ROUGE2 score obtained is 24.56 with the model run for 82 epochs.

The summaries generated has lower ROUGE-2 score compared to ROUGE-1 as it is easier to pick up on the important words in the document but because of the nature of abstractive summaries, what goes in between can be anything that makes sense for that language, ending with very low ROUGE-2 scores even when the summary seems perfectly acceptable and meaningful.. So, it is better to use ROUGE-1 metric to show fluency of the summaries over ROUGE-2 because it is difficult to follow word orderings. So to obtain concise summaries ROUGE-1 is better especially if the model applies stop word removal and stemming.

6.2. Future Recommendation

The experiment has been carried out in less amount of data and less capacity server. It would have been much better with high capacity machine and if trained with full dataset instead of mini dataset of 20000 articles. With server of higher capacity, the tuning of hyper-parameters would have been done and the result could be analyzed.

REFERNCES

[1] Bahdanau, D., Cho, K., Bengio Y., “Neural Machine Translation By Jointly Learning To Align And Translate”, ICLR 2015.

[2] C. Olah, "colah's blog," 15 8 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[3] Chen, D., Bolton, J., & Manning, C.,D., “A Thorough Examination of the CNN / Daily Mail Reading Comprehension Task”, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2358–2367,Berlin, Germany, August 7-12, 2016.

[4] Chen, V., Puzon, L., Montaño E., T., Chen, D., “An Examination of the CNN/DailyMail Neural Summarization Task”, Stanford University.

[5] Chopra, S., Auli , M., & Rush, A. M ,”Abstractive Sentence Summarization with Attentive Recurrent Neural Networks”, Proceedings of NAACL-HLT 2016, pages 93–98, 2016.

[6] D.Britz,"WILDML,"1792015.[Online]. Available:http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

[7] Deng, L., Yu, D.,”Deep Learning: Methods And Applications”, Foundations and Trends in Signal Processing, vol. 7, nos. 3–4, pp. 197–387, 2013.

[8] Feng,C., Cai,F., Chen,H., Rijke,M.,D. ” Attentive Encoder-based Extractive Text Summarization” ,International Conference on Information and Knowledge Management (CIKM ’18), October 22–26, 2018, Torino, Italy

[9] Khatri,C., Singh ,G., Parikh,N.“ Abstractive and Extractive Text Summarization using Document Context Vector and Recurrent Neural Networks”, KDD’18 Deep Learning Day, August 2018, London, UK.

[10] Kim, M.,D., Huang,C., “Neural Abstractive Summarization On Gigaword”,Standford University.

[11] Lin, C.,Y.,”Rouge: A package for automatic evaluation of summaries". Proceedings of the ACL-04 workshop. Vol. 8. Barcelona, Spain.

[12] Nallapati, R., Zhou, B., Santos, C. N. dos, Gulcehre, C., & Xiang, B. ,” Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond”, arXiv:1602.06023v5 [cs.CL] 26 Aug 2016.

[13] Nallapati,R., Xiang,B., Zhou,B. ,“Sequence to Sequence RNNs For Text Summarization”, Workshop track - ICLR 2016.

[14] Paulus,R., Xiong,C., Socher,R.,“A Deep Reinforced Model for Abstractive Summarization",arXiv:1705.04304v3 [cs.CL] 13 Nov 2017.

[15] Priya, P.,G., Duraiswamy,K.,” An Approach For Text Summarization Using Deep Learning Algorithm”, Journal of Computer Science 10 (1): 1-9, 2014.

[16] Rush,A.,M., Chopra, S., Weston,J.,” A Neural Attention Model for Abstractive Sentence Summarization”, arXiv:1509.00685v2 [cs.CL] 3 Sep 2015.

[17] Sabahi, K.,A., Zuping, Z., Kang, Y., “ Bidirectional Attentional Encoder-Decoder Model and Bidirectional Beam Search for Abstractive Summarization”, School of Information Science and Engineering, Central South University, China.

[18] Sanjabi , N., “Abstractive Text Summarization with Attention Based Mechanism”, UNIVERSITAT POLITÈCNICA DE CATALUNYA.-

[19] S. Hochreiter, J. Schmidhuber, “Long Short-Term Memory," Massachusetts Institute of Technology, pp. 1735-1780, 24 2 1997.

[20] S. Kostadinov, “Towards Data Science," 16 12 2017. [Online]. Available: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21.

[21] See,A., Liu,P.,L., Manning C.,D., “Get To The Point: Summarization with Pointer-Generator Networks”,arXiv:1704.04368v2 [cs.CL] 25 Apr 2017.

[22] Sinha,A., Yadav,A., Gahlot, A., “ Extractive Text Summarization using Neural Networks”, Department of Computer Science and Engineering Indian Institute of Technology Delhi.

[23] Singhal, S., Bhattacharya, A., “Abstractive Text Summarization”, Department of Computer Science IIT Kanpur.

[24] Sun,Q., Lee,S., Batra,D., “Bidirectional Beam Search: Forward-Backward Inference in Neural Sequence Models for Fill-in-the-Blank Image Captioning”, arXiv:1705.08759v1 [cs.CV] 24 May 2017.

[25] Yu, H., Yue, C., Wang, C., “News Article Summarization with Attention-based Deep Recurrent Neural Networks”, Standford University

[26] Young,T., Hazarika,D. Poria,S., Cambria,E., “Recent Trends in Deep Learning Based Natural Language Processing”, arXiv:1708.02709v6 [cs.CL] 4 Aug 2018.

[27] Zhu,J., Zhou,L., Li, H., Zhang, J., Zhou, Y., Zong, C., “Augmenting Neural Sentence Summarization through Extractive Summarization”, University of Chinese Academy of Sciences

[28] Cho,K.,Bahdanau,D.,Bougares,F., Schwenk,H.,Bengio,Y.,” Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”, arXiv:1406.1078v3 [cs.CL] 3 Sep 2014.

[29] Kloenne,M., “Deep recurrent neural networks for abstractive text summarization”, University Of Bielefeld, May 8, 2018.

[30] Shi,T.,Keneshloo,Y.,Ramakrishnan,N.,Reddy,C., "Neural Abstractive Text Summarization with Sequence-to-Sequence Models", arXiv:1812.02303v2 [cs.CL] 7 Dec 2018

[31] LOYE,G., "Attention Mechanism" Available: https://blog.floydhub.com/attention-mechanism/.

APPENDIX:

RNN Encoder Decoder Model With Attention:

```
class Encoder(tf.keras.Model):

  def __init__(self,
               vocab_size,
               embedding_size,
               embedding_matrix,
               lstm_size,
               pre_trained_embeddings = False,
               trainable_embeddings = True,
               dropout_rate = 0.0):

      super(Encoder, self).__init__()

      self.lstm_size = lstm_size
      self.dropout_rate = dropout_rate
      #embedding with vocabulary size as input and dimension of embeddi
ng
      self.pre_trained_embeddings = pre_trained_embeddings
      if pre_trained_embeddings:
        self.embedding = tf.keras.layers.Embedding(vocab_size,
                                                   embedding_size,
                                                   weights=[embedding_mat
rix],
                                                   trainable=trainable_em
beddings)
      else:
        self.embedding = tf.keras.layers.Embedding(vocab_size,
                                                   embedding_size,
                                                   trainable=trainable_em
beddings)


      self.backward_layer = tf.keras.layers.LSTM(lstm_size // 2, return
_sequences=True,
                                                 return_state=True, go_b
ackwards=True)
      self.forward_layer = tf.keras.layers.LSTM(lstm_size // 2, return_
sequences=True,
                                                return_state=True)

      self.bidirectional = tf.keras.layers.Bidirectional(layer = self.f
orward_layer,
                                                         backward_layer
= self.backward_layer,
```

```
                                                        merge_mode = '
concat')
      self.dropout = tf.keras.layers.Dropout(rate = dropout_rate)


  def call(self, sequence, states, training_flag = False):
      embed = self.embedding(sequence)
      embed_dropout = self.dropout(embed, training = training_flag )
      outputs = self.bidirectional(embed_dropout, initial_state=states)
      output = outputs[0]
      state_f_h = outputs[1]
      state_f_c = outputs[2]
      state_b_h = outputs[3]
      state_b_c = outputs[4]
      state_h = tf.concat([state_f_h, state_b_h],-1)
      state_c = tf.concat([state_f_c, state_b_c],-1)
      return output, state_h, state_c

  def init_states(self, batch_size):

      return [tf.zeros([batch_size, self.lstm_size // 2]),
              tf.zeros([batch_size, self.lstm_size // 2]),
              tf.zeros([batch_size, self.lstm_size // 2]),
              tf.zeros([batch_size, self.lstm_size // 2])]

# Luong Attention subclass

class LuongAttention(tf.keras.Model):
    def __init__(self, rnn_size):
        super(LuongAttention, self).__init__()
        self.wa = tf.keras.layers.Dense(rnn_size)

    def call(self, decoder_output, encoder_output):
        score = tf.matmul(decoder_output, self.wa(encoder_output), tran
spose_b=True)
        alignment = tf.nn.softmax(score, axis=2)
        context = tf.matmul(alignment, encoder_output)

        return context, alignment

## Decoder subclass

class Decoder(tf.keras.Model):

  def __init__(self,
               vocab_size,
               embedding_size,
               embedding_matrix,
```

```
                lstm_size,
                pre_trained_embeddings = False,
                trainable_embeddings = True):

      super(Decoder, self).__init__()
      self.attention = LuongAttention(lstm_size)
      self.lstm_size = lstm_size
      self.pre_trained_embeddings = pre_trained_embeddings
      if pre_trained_embeddings:
        self.embedding = tf.keras.layers.Embedding(vocab_size,
                                                   embedding_size,
                                                   weights=[embedding_mat
rix],
                                                   trainable=trainable_em
beddings)
      else:
        self.embedding = tf.keras.layers.Embedding(vocab_size,
                                                   embedding_size,
                                                   trainable=trainable_em
beddings)
      self.lstm = tf.keras.layers.LSTM(
          lstm_size, return_sequences=True, return_state=True)
      self.wc = tf.keras.layers.Dense(lstm_size, activation='tanh')
      self.ws = tf.keras.layers.Dense(vocab_size)

  def call(self, sequence, state, encoder_output):
      embed = self.embedding(sequence)
      lstm_out, state_h, state_c = self.lstm(embed, initial_state = sta
te)
      context, alignment = self.attention(lstm_out, encoder_output)
      lstm_out = tf.concat([tf.squeeze(context, 1), tf.squeeze(lstm_out
, 1)], 1)

      # lstm_out now has shape (batch_size, rnn_size)
      lstm_out = self.wc(lstm_out)

      # Finally, it is converted back to vocabulary space: (batch_size,
 vocab_size)
      logits = self.ws(lstm_out)
```