Shahjalal University of Science and Technology

Department of Computer Science and Engineering

Bengali Text Summarizer Project

Sujoy Nath

Reg. No.: 2014331050

4^{th} year, 1^{st} Semester

Shahnur Islam

Reg. No.: 2014331072

4^{th} year, 1^{st} Semester

Department of Computer Science and Engineering

Supervisor

Md Mahfuzur Rahaman

Assistant Professor

Department of Computer Science and Engineering

July 18, 2018

Recommendation Letter from Thesis Supervisor

This project is entitled as ” Bengali Text Summarization” submitted by the students

1. Sujoy Nath
2. Shahnur Islam

is a record of research work carried out under my supervision and I, hereby, approve that the report be submitted in partial fulfillment of the requirements for the award of their Bachelor Degrees.

Signature of the Supervisor:
Name of the Supervisor: Md Mahfuzur Rahaman
Date: July 18, 2018

Certificate of Acceptance of the Thesis

The project is
entitled ”Bengali Text Summarizer” submitted by the students

1. Sujoy Nath
2. Shahnur Islam

on July 17, 2018
as part of the requirements of the course CSE-450, is being approved by the Department of Computer Science and Engineering as a partial fulfillment of the B.Sc.(Engg.) degree of the above students.

Supervisor Md Mahfuzur Rahaman Assistant Professor Department of Computer Science and Engineering	Head of the Dept. Dr Mohammad Reza Selim Professor Department of Computer Science and Engineering	Chairman, Exam. Committee Dr Farida Chowdhury Associate Professor Department of Computer Science and Engineering

Abstract

Text Summarization is the process of creating a condensed form of text document which maintains significant information and general meaning of source text. Automatic text summarization becomes an important way of finding relevant information precisely in large text in a short time with little efforts. We can find a lot of works have been done in English text summarization but a very few attempts have been made for Bengali text summarization. The number of Bengali data is increasing day by day and these data are not preserved in a structured way. We can preserve these data by creating summary from it's original document. There are two types of text summarization extractive and abstractive. LSTM is a deep learning process to generate an abstractive text summarization and TF-IDF is the process of generating extractive text summarization. In this project we have worked for both extractive and abstractive summarization. For extractive summarization we have used TF-IDF and TextRank and for abstractive summarization we have used an encoder-decoder LSTM model with attention mechanism. For extractive summarization we have used Bengali online newspaper's data and we have generated a summarizer for Bengali document. But by the extractive text summarization we can not generate the actual summary of large document. Further we have tried for English abstractive summarizer. For abstractive summarizer we have used the dataset from the news collection by the Associated Press Worldwide from English Gigaword, a dataset of historical news articles and headlines from five reputable, international news agencies.[1] We have not got our expected output from the abstractive summarization because we have not trained our data properly for the lacking of our computer configuration. Further we will train our data for both Bengali and English language which will provide us the abstractive Bengali text summarizer.

Keywords :TF-IDF, LSTM, Encoder-Decoder Model, Attention Machanism

Acknowledgements

First of all, we would like to give thanks from the core of our heart to our honurable advisor Md Mahfuzur Rahaman, Assistant Professor, Department of CSE, SUST, for the constant support to us. His continuous direction to literature study and experiment helps us to conduct our project.

We also want to express our gratitude to previous authors and researchers for their direct and indirect contribution in this project.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

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 is where the machine takes actual context and uses that as the summary.[1] It is the equivalent of copying down the main points of a text without any changes. Abstractive is a technique used to mimic paraphrasing. This method makes summarization more condensed and human like.[1]

As a deep learning problem, writing good summaries is a noteworthy task. We will explore ways to implement a recurrent neural network to automatically generate summary that summarizes the main idea from the text of the news articles.

We believe that our work is significant because we design our user studies to prioritize human significance rather than simply mathematical significance. We have come away from this semester with a deeper, hands-on experience with using commercial cloud computing products like, Google's TensorFlow, one of many machine learning and deep learning frameworks publicly available

1.1 Background

Day by day quantity of electronic data is increasing exponentially. As data increased rapidly, summarizing data using deep learning is not a worthy task ..

Lots of research has been done in last few decades to summarize text . Recurrent Neural Network has been worked so far successfully to summarize sequence to sequence texts, images , videos etc . One of the google research team first use Recurrent Neural Network to produce to summarize text

As we have to generate a sequence of data from sequence of input text that's why we are using Recurrent Neural Network for text summarization. In this project we have used supervised deep learning approach to summarize text.

Bengali is one of the most popular language in the world. It is in 7th position in the list of most spoken languages. As the number of Bengali native speaker increased, elec-tronic documents written in Bengali language also increased. So, the summary of Bengali documents will be desirable.

1.2 Motivation

There are many reasons and uses for a summary of a larger document.One example that might come readily to mind is to create a concise summary of a long news article, but there are many more cases of text summaries that we may come across every day. Here are some useful list of every-day examples of text summarization.

- headlines (from around the world)
- outlines (notes for students)
- minutes (of a meeting)
- previews (of movies)
- synopses (soap opera listings)
- reviews (of a book, CD, movie, etc.)
- digests (TV guide)
- biography (resumes, obituaries)
- abridgments (Shakespeare for children)
- bulletins (weather forecasts/stock market reports)
- sound bites (politicians on a current issue)
- histories (chronologies of salient events)

1.3 Goal

Creating a concise summary for Bengali document .So far we have done a term-frequency based extractive summarizer and have analyzed some model of text summarization based on English document.

Chapter 2

Background Study

2.1 Literature Review

Before starting the research we review some papers that work on Document Categorization. Most of them have used English language documents for their research purpose. Few researches are also performed for other languages.

Throughout our literature review we marked some techniques that are used in the most of the researches. They are word to vect, glove vector ,RNN ,LSTM ,attention distributor ,context vector ,k-mean ,TF-IDF Some of them are described in the following section.

2.1.1 **WordToVect**

Word2vec is a two-layer neural net that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus. While Word2vec is not a deep neural network, it turns text into a numerical form that deep nets can understand. The output of the Word2vec neural net is a vocabulary in which each item has a vector attached to it, which can be fed into a deep-learning net or simply queried to detect relationships between words.[7]

Measuring cosine similarity, no similarity is expressed as a 90 degree angle, while total similarity of 1 is a 0 degree angle, complete overlap; i.e. Sweden equals Sweden, while Norway has a cosine distance of 0.760124 from Sweden, the highest of any other country.[7]

Here’s a list of words associated with “Sweden” using Word2vec, in order of proximity

Table 1: Word to vect table for cosine distance of particular words

Word	Cosine Distance
Norway	0.760124
Denmark	0.715460
Finland	0.620022
Switzerland	0.588132
Belgium	0.585835
Iceland	0.562368

2.1.2 Glove Vector

GloVe, coined from Global Vectors, is a model for distributed word representation. The model is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.[8]

It generates a large vector using context of the text as number of the vector is very large so it converted that vector into lower dimension by factorization which represents as features and words. This representation of glove vector is known as word embedding

2.1.3 **Recurrent Neural Network(RNN)**

Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many NLP tasks. The idea behind RNNs is to make use of sequential information. [3]

In a traditional neural network we assume that all inputs (and outputs) are independent of each other. But for many tasks that's a very bad idea. If you want to predict the next word in a sentence you better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations.[3]

Here is what a typical RNN looks like:

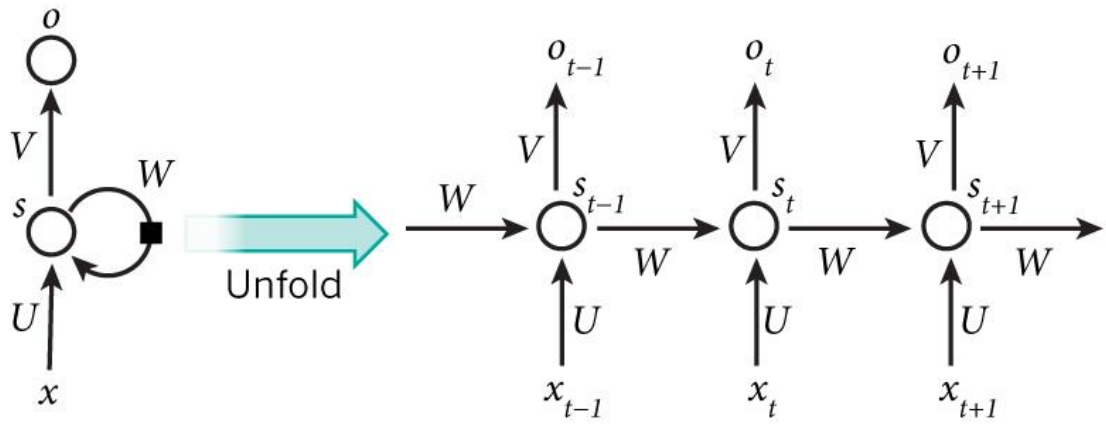


Figure 2.1: Typical Neural Network [3]

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

- x_t is the input at time step t. For example, x_1 could be a one-hot vector corresponding to the second word of a sentence.
- s_t is the hidden state at time step t. It's the "memory" of the network. 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 ReLU. s_{-1}, which is required to calculate the first hidden state, is typically initialized to all zeroes.
- o_t is the output at step t. For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary. $o_t = \mathrm{softmax}(Vs_t)$.[3]

2.1.3 **Long Short Term Memory (LSTM)**

Different short-term memories should be recalled at different times, in order to assign the right meaning to current input in the hidden state of RNN. Some of those memories will have been forged recently, and other memories will have been forged many time steps before they are needed. The recurrent net that effectively associates memories and input remote in time is called a Long Short-Term Memory (LSTM).[9]

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn![9]

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.[9]

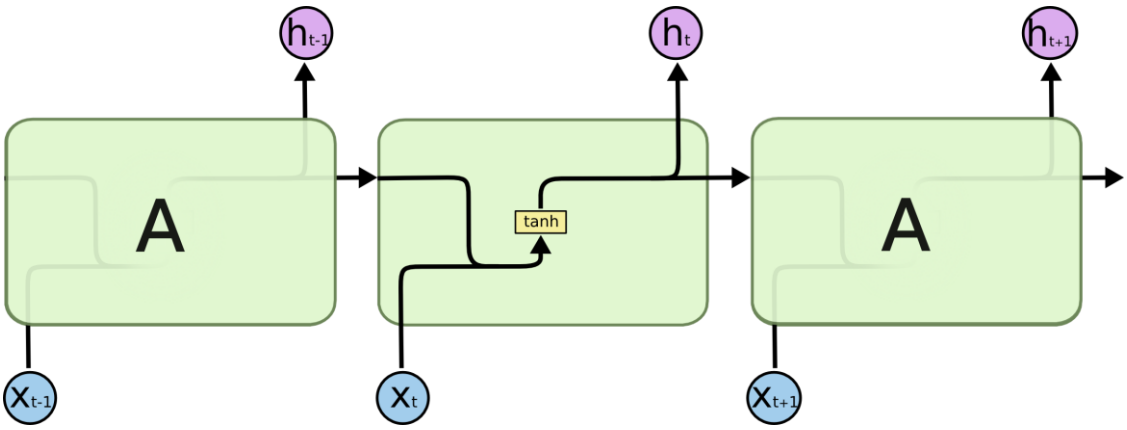


Figure 2.2: The repeating module in a standard RNN contains a single layer.

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.[9]

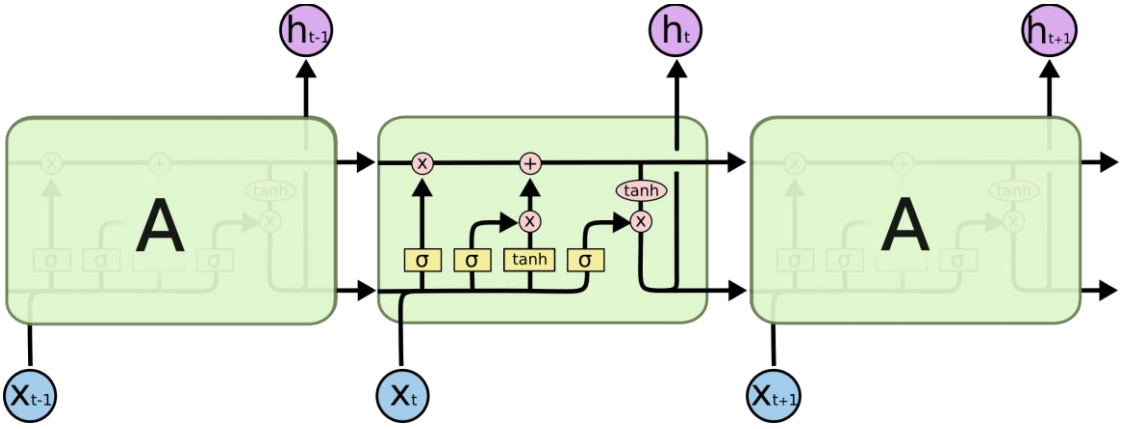


Figure 2.3 The repeating module in an LSTM contains four interacting layers.

2.1.4 **Attention Distributor and Context Vector**

Attention is simply a vector, often the outputs of dense layer using softmax function. Before Attention mechanism, translation relies on reading a complete sentence and compress all information into a fixed-length vector, as you can image, a sentence with hundreds of words represented by several words will surely lead to information loss, inadequate translation, etc.[6]

However, attention partially fixes this problem. It allows machine translator to look over all the information the original sentence holds, then generate the proper word according to current word it works on and the context. It can even allow translator to zoom in or out (focus on local or global features). (focus on local or global features).[6]

Attention is not mysterious or complex. It is just an interface formulated by parameters and delicate math. You could plug it anywhere you find it suitable, and potentially, the result may be enhanced.[6]

The attention distribution is used to produce a weighted sum of the encoder hidden states, known as the context vector. The context vector can be regarded as “what has been read from the source text” on this step of the decoder[6].

Finally, the context vector and the decoder hidden state are used to calculate the vocabulary distribution, which is a probability distribution over all the words in a large fixed vocabulary (typically tens or hundreds of thousands of words). The word with the largest probability (on this step, beat) is chosen as output, and the decoder moves on to the next step.[6]

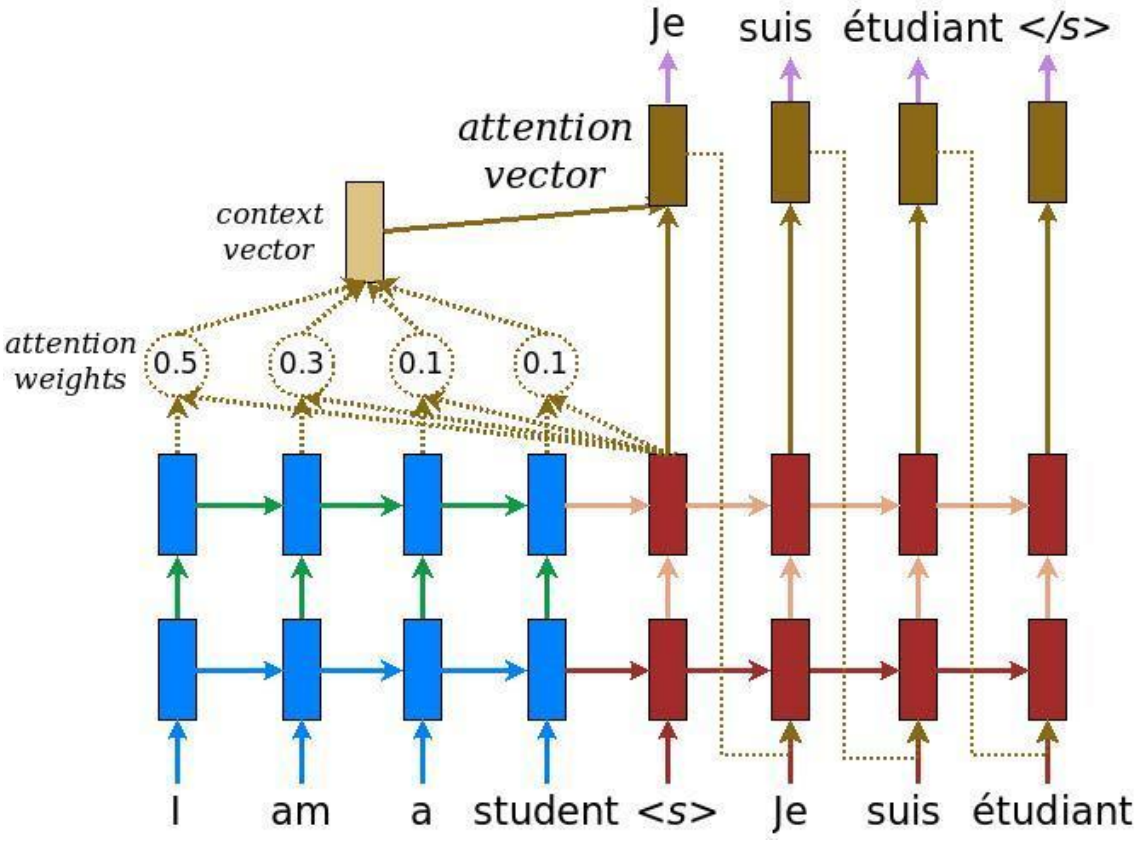


Figure 2.4 Attention vector and context vector model

2.1.5 K-Means Clustering Algorithm

k-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed apriori. [5]

The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other.[5]

The next step is to take each point belonging to a given data set and associate it to the nearest center. When no point is pending, the first step is completed and an early group age is done. At this point we need to re-calculate k new centroids as barycenter of the clusters resulting from the previous step.[5]

After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new center. A loop has been generated. As a result of this loop we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more. Finally, this algorithm aims at minimizing an objective function know as squared error function given by: [5].

$$J(V) = \sum_{i=1}^{c} \sum_{j=1}^{c_i} \left(\left\| x_i - v_j \right\| \right)^2$$

where,

'$||x_i - v_j||$' is the Euclidean distance between x_i and v_j.

'c_i' is the number of data points in i^{th} cluster.

'c' is the number of cluster center

Chapter 3

Existing Methodology

For document categorization there are so many methodology exist. Some of them are impressive and some of them aren't enough good. In this chapter we will discuss few of them.

3.1 Sequence to Sequence Model Using Keras

In this section, we will look at how to implement the Encoder-Decoder architecture for text summarization in the Keras deep learning library.

General Model

A simple realization of the model involves an Encoder with an Embedding input followed by an LSTM hidden layer that produces a fixed-length representation of the source document.

The Decoder reads the representation and an Embedding of the last generated word and uses these inputs to generate each word in the output summary.[4]

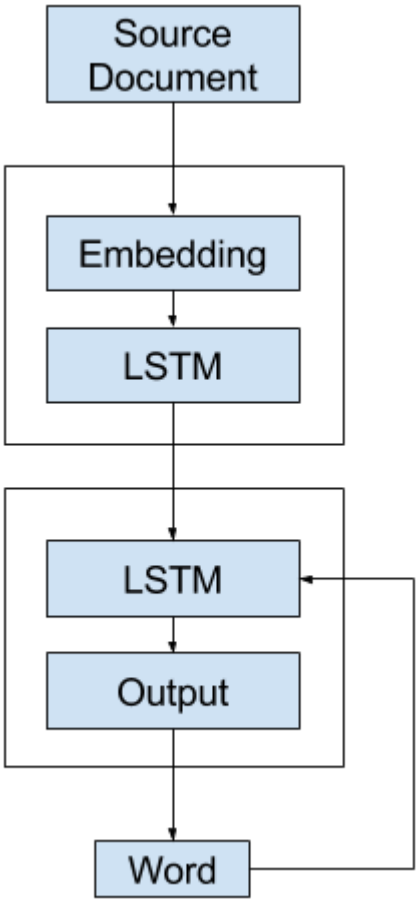


Figure 3.1 sequence to sequence keras model-1

There is a problem. Keras does not allow recursive loops where the output of the model is fed as input to the model automatically.This means the model as described above cannot be directly implemented in Keras (but perhaps could in a more flexible platform like TensorFlow).[4]Instead, we will look at three variations of the model that we can implement in Keras.

Alternate 1: One-Shot Model

The first alternative model is to generate the entire output sequence in a one-shot manner.That is, the decoder uses the context vector alone to generate the output sequence.[4]

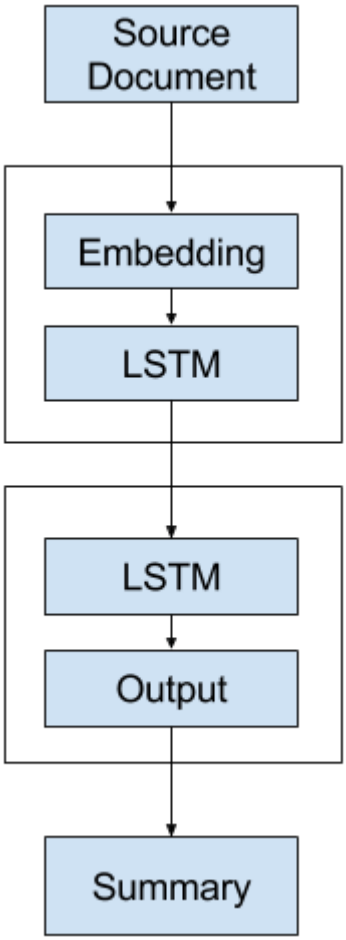


Figure 3.2 sequence to sequence keras model-2

This model puts a heavy burden on the decoder.It is likely that the decoder will not have sufficient context for generating a coherent output sequence as it must choose the words and their order.[4]

Alternate 2: Recursive Model A

A second alternative model is to develop a model that generates a single word forecast and call it recursively.That is, the decoder uses the context vector and the distributed representation of all words generated so far as input in order to generate the next word.[4

A language model can be used to interpret the sequence of words generated so far to provide a second context vector to combine with the representation of the source document in order to generate the next word in the sequence.[4]

The summary is built up by recursively calling the model with the previously generated word appended (or, more specifically, the expected previous word during training).[4]

The context vectors could be concentrated or added together to provide a broader context for the decoder to interpret and output the next word

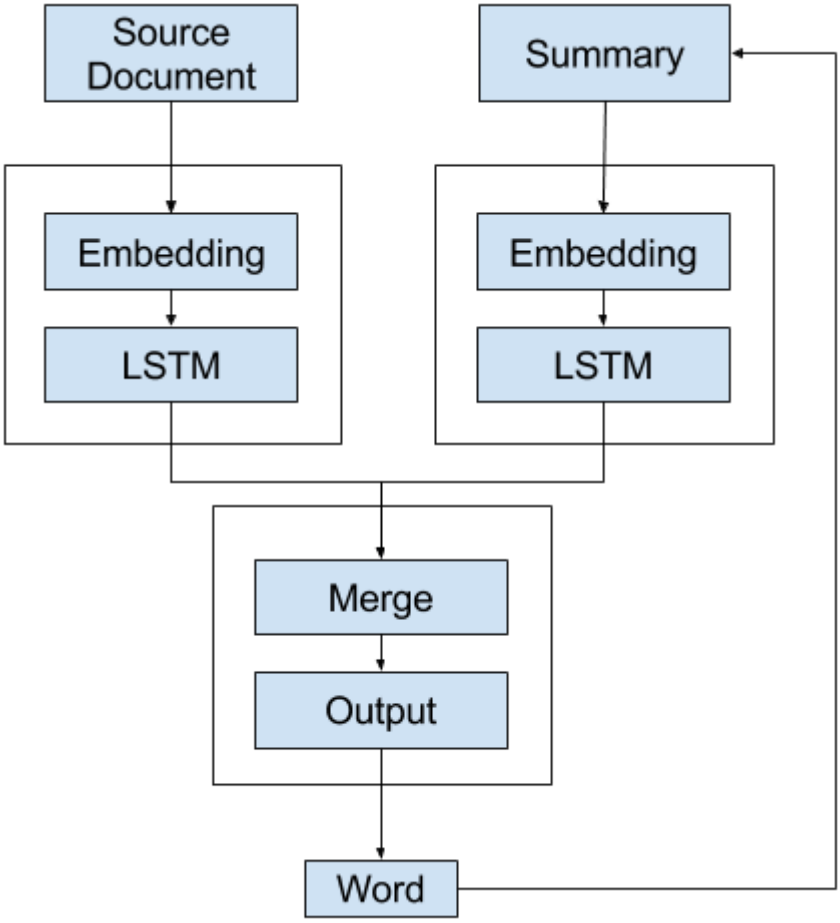


Figure 3.3 sequence to sequence keras model-3

Sequence to Sequence RNN

sequence-to-sequence is a *problem setting*, where your input is a sequence and your output is also a sequence. Typical examples of sequence-to-sequence problems are machine translation, question answering, generating natural language description of videos, automatic summarization, etc. LSTM and RNN are *models*, that is, they can be trained using some data and then be used to predict on new data. You can have *sequence-to-sequence models* that use LSTMs/RNNs as modules inside them, where a “sequence-to-sequence model” is just a model that works for sequence to sequence tasks.[10]

RNNs is a class of neural networks that is not feed-forward. A feed-forward neural network looks something like this:

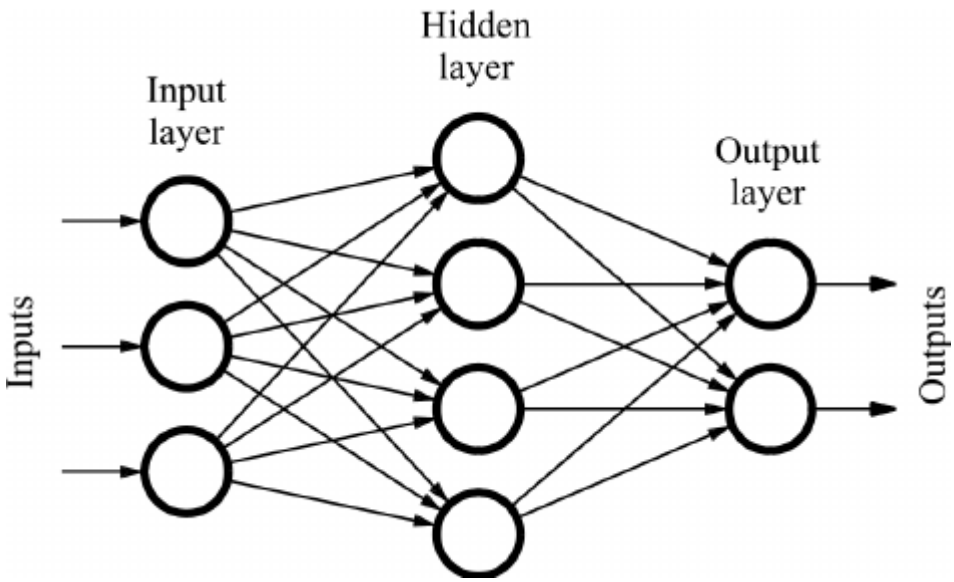
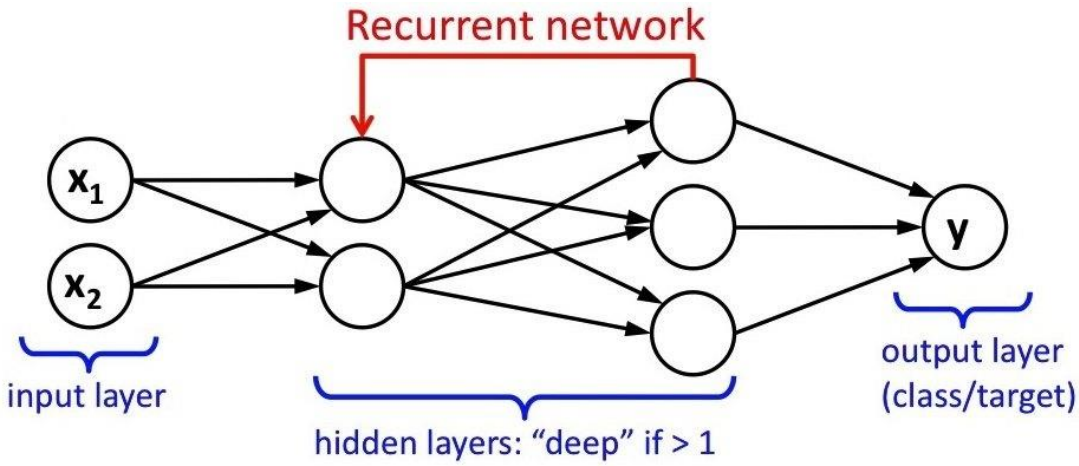


Figure 3.4 Sequence to sequence model

As the name suggests, you only feed the input in the forward direction, that is, the output from one unit is only fed to units further ahead.[10]

On the other hand, an RNN looks something like the following:



That is, there are loops in the network graph, and the output of one unit may go back to one of the already visited units.[10]

Finally, LSTMs are a special type of RNNs, where you connect these units in a specific way, to avoid some problems that arise in regular RNNs [vanishing and exploding gradient][10]

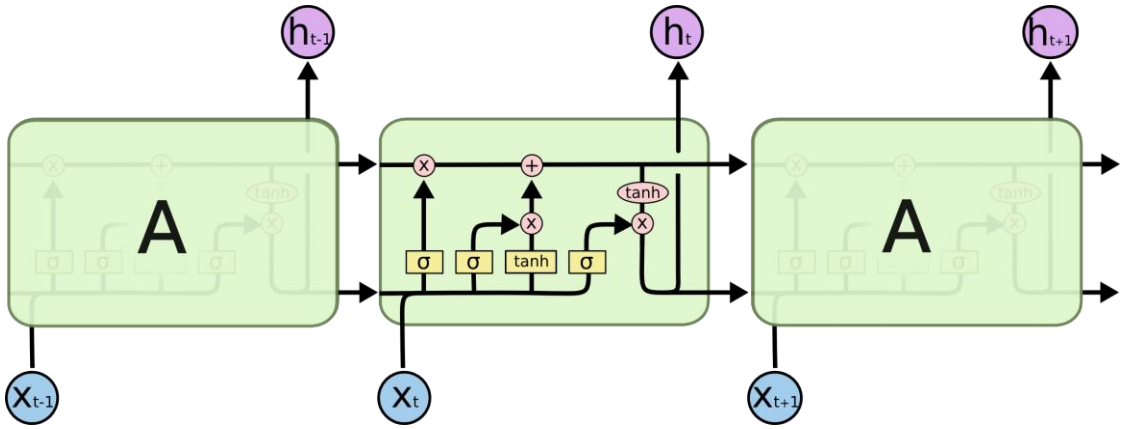


Figure 3.5 Recurrent Neural Network

Here, A is the unit that takes an input X, outputs a vector h, and also some other vectors that are fed back to A [shown by right arrows from one A to another]. Note that all the A’s here refer to a single object in memory — they are drawn in this unrolled way for explanation. What’s really happening is that the outputs that come out from the right of A are fed back from the left to the same A module.[10]

There are many ways in which you can use LSTMs for sequence-to-sequence tasks. As the above diagram shows, you can feed in the input sequence to an LSTM and get a sequence of h vectors. You can then process these h vectors further.

Chapter 4

Proposed Methodology

Developing a Bengali Text Summarizer is our main concern . So far we have analysed some models of English Text Summarizer . Considering these model we are developing Bengali Summarizer using sequence to sequence recurrent neural network as rnn performs better for sequential data.

4.1 LSTM Encoder Decoder

First, the encoder RNN reads in the source text word-by-word, producing a sequence of encoder hidden states. Once the encoder has read the entire source text, the decoder RNN begins to output a sequence of words that should form a summary. On each step, the decoder receives as input the previous word of the summary (on the first step, this is a special <START> token which is the signal to begin writing) and uses it to update the decoder hidden state.[2]

This is used to calculate the attention distribution which is a probability distribution over the words in the source text. Intuitively, the attention distribution tells the network where to look to help it produce the next word Next, the attention distribution is used to produce a weighted sum of the encoder hidden states, known as the context vector. The context vector can be regarded as “what has been read from the source text” on this step of the decoder.[2]

Finally, the context vector and the decoder hidden state are used to calculate the vocabulary distribution, which is a probability distribution over all the words in a large fixed vocabulary (typically tens or hundreds of thousands of words).[2]

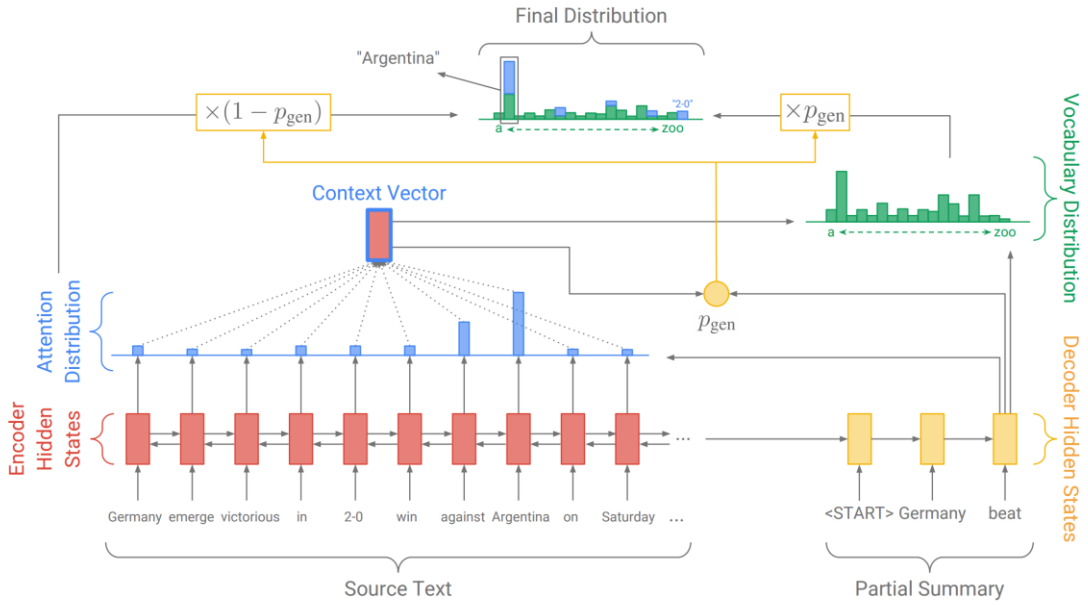


Figure 4.1: Encoder-decoder LSTM model with attention mechanism

We use the encoder-decoder LSTM model with attention mechanism to generate a headline given the first sentence of a news article. in the sentences while [1] only use one bucket. Second, we use a greedy decoder rather then a beam search decoder to generate headline words.An encoder-decoder LSTM model with attention is shown in Figure 4.1. [2]

We used TensorFlow library to implement our encoderdecoder LSTM model with attention mechanism. Our code is build from both sequence-to-sequence model2 from TensorFlow. Specifically, we use three LSTM hidden layers in both encoder and decoder model, an output dropout rate of 0.2 between layers, 512 hidden units per LSTM cell, and we trained word embeddings to 512 dimensions. We embedded the most frequent 80,000 words in both sentences and headlines, and label all other words as unknown words (UNK) or often known as out-of-vocabulary (OOV) words.[1]

Lastly, we use five buckets ((30, 10), (30, 20), (40, 10), (40, 20), (50, 20)) (i.e, five different encoder-deocder LSTM model) based on the statistics acquired in section 3.3 to reduce the number of padding tokens. In bucket representation (a, b), a refers to the length of the sentence and the b is the length of headline. If a sentence is longer then 50 tokens or a headline longer then 20 tokens, we automatically place it in the last ((50,20)) bucket. We use a special token (PAD) to pad both the sentences and headlines to its bucket size[1]

Chapter 5

Data Sets

The model is trained using the English Gigaword 2nd edition dataset, as available from the University of Texas library. This dataset consists of several years of newswire text data from five major domestic and international news services: Agence France-presse English Service, Associated Press Worldstream English Service (APW), the New York Times Newswire Service (NYT), Central News Agency of Taiwan English Service, and the Xinhua News Agency English Service[1]

5.1 Data Sources

Currently we are using dataset of Gigaword 2nd edition dataset, as available from the University of Texas library for English summarizer[1]

To increase the efficiency, we need to train our system with huge amount of data. So we will collect more Bengali documents from various sources for our future Bengali summarizer

We will use following sources for collecting Bengali documents:

- Prothom-Alo Online Newspaper
- BDNews24
- Somewhereinblog.net
- RoarBangla Media

5.3 Data Preprocess

This is the process which we will call data preprocessing. In its raw form, the English Gigaword dataset contains hundreds of compressed files distributed across five directories for the five news organizations. Each compressed file contains on the order of 10,000 articles in Standard Generalized Markup Language (SGML) format, an HTML-like markup language. Sections in each datum include: document metadata (unique ID and document type), the headline of the news article, and multiple paragraphs of text which comprise the article,[1]

We use a markup language parsing Python library called BeautifulSoup4. This enables us to parse the data in a way that is flexible to our needs .We further truncated our dataset based on a number of criteria which availed themselves throughout the preprocessing and training process.[1]

Chapter 6

Experiments and Results

6.1 **Term Frequency based extractive summarization**

In this approach we took a large document and produce a few line of summary without changing the text from original document. First from a large document we remove stop words and tokenize document into words and sentences . Then we use steamer to find out the root word and consider the term frequency for document .

Then using TextRank algorithom we generate a summary from original document.
Here is an example of our Bengali extractive summarizer

Input text:

লিওনেল মেসি, না হয় ক্রিস্টিয়ানো রোনালদো , বছরের নির্ধারিত সময়ে যেকোনো একজনের মাথাতেই ওঠে ফিফা ব্যালন ডি'অরের মুকুট। দৃশ্যটা যেন একেবারেই সাজানো চিত্রনাট্যের মতো হয়ে উঠেছে। কম তো হলো না, সেই ২০০৮ সাল থেকে পাল্লা দিয়ে চলছে দুজনের বিশ্বসেরা ফুটবলার হওয়ার দৌড়। মেসির নামের পাশে পাঁচটি, রোনালদোর নামের পাশেও তা-ই ,এবার কি অন্য রকম কিছু হতে চলেছে!

হয়তো! যদিও এখনো রোনালদোর দিকে পাল্লা অনেক ভারী, তবে বিশ্বকাপের বছরে এই টুর্নামেন্টের সাফল্য অনেকটাই এগিয়ে দেয় এই আসরের সফলদের। ফিফা বর্ষসেরার সঙ্গে আলাদা হয়ে যাওয়ায় এই পুরস্কার ভোটাভুটির অধিকার আবার শুধু সাংবাদিকদের কাছে থাকল। এ কারণে অনেকে মনে করছেন, ব্যালন ডি'অর এবার এমবাপ্পের কাছে যেতে পারে।

Output Text

লিওনেল মেসি, না হয় ক্রিস্টিয়ানো রোনালদো , বছরের নির্ধারিত সময়ে যেকোনো একজনের মাথাতেই ওঠে ফিফা ব্যালন ডি'অরের মুকুট। মেসির নামের পাশে পাঁচটি, রোনালদোর নামের পাশেও তা-ই ,এবার কি অন্য রকম কিছু হতে চলেছে! হয়তো! যদিও এখনো রোনালদোর দিকে পাল্লা অনেক ভারী, তবে বিশ্বকাপের বছরে এই টুর্নামেন্টের সাফল্য অনেকটাই এগিয়ে দেয় এই আসরের সফলদের।

6.2 Sequence to sequence RNN model :

We Followed Headline Generation Using Recurrent Neural Networks paper and their methodology.

They splitted the 1.3 million articles and headlines pairs acquired from 1994-2004 APW newswire into 70% training set, 20% evaluation set, and 10% testing set. They used batched training of size 128, a learning rate of 0.5 using gradient descent optimizer with a decay factor of 0.99 and set the maximum gradient norm to 5. Then they trained their sequence to sequence model on AWS EC2 g2.2xlarge (NVIDIA K520 with 15G memory) instance for 10 epochs, which took around 20 hours. They had got some awesome output from their model. Here is given a table of their Input and output data [1]

Table 1 : Examples of predicted headlines

Sentence	Actual Headlines	Predicted Headlines
President Vladimir Putin announced plans Monday for the russain military to hold exercise in the former soviet republic of Kyrgyzstan the itar-tass news agency reported	Russain Plans military exercise in Kyrgyzstan	Putin to launch military exercise in Kyrgyzstan

We have tried this to our own computer. As our computer configuration is not enough for this train procedure so we were unable to train enough data and we did not get our expected result for all input tests. Here is an example of our own test

Table 2: Examples of predicted headlines

Sentence	Actual Headlines	Predicted Headlines
President Vladimir Putin announced plans Monday for the russain military to hold exercise in the former soviet republic of Kyrgyzstan the itar-tass news agency reported	Russain Plans military exercise in Kyrgyzstan	President president former Kyrgyzstan

Chapter 7

Conclusion

Lots of work have been done so far for English Text Summarizer but still we do not find any Bengali summarizer which can summarize Bengali document. So using the above methodology we will develop a abstractive Bengali summarizer to concise text from a large text.

In this report, we try to explain the background and emphasize of our project, what types of works have been done so far by others for automatic english text summarizer and how we developed our extractive term-frequency based summarizer.

First few months of our research we spent time to read some research papers that are written on this topic so that we can acquire some ideas of currently going trend and how they had done their experiment. In the literature review section of this report we mention some of our studied papers. Further we also analyzed some models of extractive and abstractive text summarization in English and also developed a term frequency based extractive Bengali summarizer model successfully.

.

7.1 Future Work

Our aim is to develop an automatic abstractive Bengali text summarizer. Still we are far way from our goal. In future we will develop a abstractive Bengali text summarizer

7.1.1 Data Collection

We do not have enough Bengali data for our text summarization project. We will collect more data to ensure efficiency of our project

References

[1] Heng-Lu Chang, Yung-Shen Chang, Linli Ding, Jeremy Gin Headline Generation Using Recurrent Neural Networks

[2] Abigail See, Peter J. Liu, Christopher D. Manning Get To The Point: Summarization with Pointer-Generator Networks

[3] Wildmil Artificial Intelligence, Deep Learning, and NLP blog

[4] Jason Brownlee Encoder-Decoder Models for Text Summarization in Keras from Machine Learning Mastery

[5] Data Clustering Algorithm google site

[6] A Brief Overview of Attention Mechanism – SyncedReview – Medium

[7] Word2Vec, Doc2vec & GloVe: Neural Word Embeddings for Natural Language Processing

[8] Wikipedia

[9] Understanding LSTM Networks of colah's blog

[10]https://www.quora.com/What-is-the-difference-between-LSTM-RNN-and-sequence-to-sequence