Comparative Study of Text Summarization Algorithms

Team Members :
Nanda Kishore Kasula
Shireen Firdoz

Course : CS6120
Advisor : Prof. Ahmad Uzair

Abstract : Text Summarization is basically creating a summary of long text by extracting core ideas. Extractive Text Summarization is to take up a document and extract important sentences from it as it is without changing any word.The algorithm based on ranking the sentences is based on TextRank and Word Frequency. Abstractive Text Summarization is creating a summary from a document depending on the semantics of the document and not the exact sentences, and we achieved it by using the Seq2Seq Model(LSTM). We chose two data set one is **Amazon Review** data set, which is a small size length text comprises of reviews shared by amazon customer about their products and second is **CNN Daily Mails,** which is medium size length mails and trained three models (TextRank, Word Frequency, Seq2Seq) on both the dataset to compare the results (summary generated) evaluated through rouge score.

Introduction : Information available in today's world is in abundance and a lot of new possibilities can be explored from them. Text summarization is one of the main applications of natural language processing. Text summarization is one of the widely used methods to process the text corpus and obtain a precise text that captures the entire context and preserves the important information conveyed through the text. Some use cases where automatic summarization can be used across the enterprise are Search Marketing and SEO, Financial Research, Question answering and bots, Video Scripting, Patent research and many more.

Goal : Comparative Study of Text Summarization Algorithms using models based on TextRank, Word Frequency and Seq2Seq LSTM.

Data-Set :

a) Amazon Review Data Set
(https://www.kaggle.com/code/currie32/summarizing-text-with-amazon-reviews/notebook)
b) CNN Daily Mails
(https://www.kaggle.com/datasets/gowrishankarp/newspaper-text-summarization-cnn-dailymail)

Code Link :

Following is the link to Github Repository:
https://github.com/nandakishorek1995/CS-6120-Final-Project

Method : We implemented the Extractive Text Summarization using two approaches one is using the Term Frequency -Inverse Document Frequency (TF-IDF) and the other one using Text Rank Algorithm. Apart from these two extractive approaches we have also implemented an Abstractive Text Summarization using Sequence 2 Sequence LSTM model.

Term Frequency - Inverse Document Frequency (TF-IDF)
TFIDF, short for term frequency–inverse document frequency, is a numerical measure that is used to score the importance of a word in a document based on how often it appears in that document and a given collection of documents. The intuition behind this measure is : If a word appears frequently in a document, then it should be important and we should give that word a high score. But if a word appears in too many other documents, it's probably not a unique identifier, therefore we should assign a lower score to that word.

Formula for calculating tf and idf:

- **TF(w)** = (Number of times term w appears in a document) / (Total number of terms in the document)
- **IDF(w)** = log_e(Total number of documents / Number of documents with term w in it)

Hence tfidf for a word can be calculated as:

$$\mathbf{TFIDF(w) = TF(w) * IDF(w)}$$

The value of TF-IDF ranges from zero to one with ten-digit precision.After knowing the TF-IDF value of each word, it can calculate the score of the sentence. The score of a sentence is a sum of the value of every word in the sentence. The score of the sentence gives weight to the paragraph.

Finally, three to five sentences with the TF-IDF value are chosen which have the sentence score more than the average score of the sentences.. As TF-IDF is an extraction method, the sentences that appear in the summary are the same as the original document. These chosen final sentences are sorted in accordance with its appearance in the original document. For the multi-document summarization, the sentences are sorted similarly with single document summarization. The difference is that it starts from the document which has the lowest total of TF-IDF.

Text Rank based Algorithm
The algorithm Text Rank is a graph-based ranking algorithm for text processing which is used in order to find the most relevant sentences in text and also to find the keywords. Text Rank algorithm is based on the Page Rank method which is often used in ranking of the websites , keyword extraction and text summarization. Before unfolding Text Rank let's first understand Page Rank and intuition behind it:

PageRank assumes that the rank of a webpage W depends on the importance of a webpage suggested by other web pages in terms of links to the page i.e if a webpage 'X' has a link to webpage 'W', 'X' contributes to the importance of 'W'.

For example, suppose we have four web pages namely; w1, w2, w3, and w4. These pages contain links pointing to one another. In which some pages might have no link; these are called dangling pages.

webpage	links
w1	[w4, w2]
w2	[w3, w1]
w3	[ ]
w4	[w1]

- Web page w1 has links directing to w2 and w4
- w2 has links for w3 and w1
- w4 has links only for the web page w1

In order to rank these pages, we would have to compute a score called the PageRank score. This score is the probability of a user visiting that page.

To capture the probabilities of users navigating from one page to another, we will create a square matrix M, having n rows and n columns, where n is the number of web pages.

$$M = \begin{array}{c|c|c|c|c|} & \text{w1} & \text{w2} & \text{w3} & \text{w4} \\ \hline \text{w1} & & & & \\ \hline \text{w2} & & & & \\ \hline \text{w3} & & & & \\ \hline \text{w4} & & & & \\ \hline \end{array}$$

Each element of this matrix denotes the probability of a user transitioning from one web page to another.For ex, the highlighted cell below contains the probability of transition from w1 to w2.

$$M = \begin{array}{c|c|c|c|c|} & \text{w1} & \text{w2} & \text{w3} & \text{w4} \\ \hline \text{w1} & & \blacksquare & & \\ \hline \text{w2} & & & & \\ \hline \text{w3} & & & & \\ \hline \text{w4} & & & & \\ \hline \end{array} \qquad \textbf{P(w1 to w2)}$$

The initialization of the probabilities is explained in the steps below:

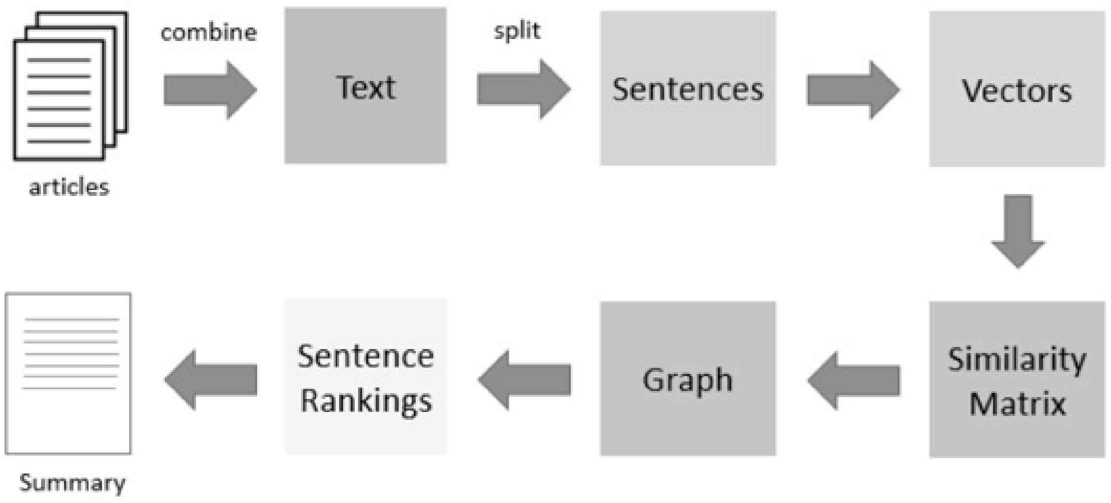
1. Probability of going from page i to j, i.e., M[i][j], is initialized with 1/(number of unique links in web page wi)
2. If there is no link between the page i and j, then the probability will be initialized with 0
3. If a user has landed on a dangling page, then it is assumed that he is equally likely to transition to any page. Hence, M[i][j] will be initialized with 1/(number of web pages)

The Matrix M will be initialized with values as given below:

$$M = \begin{array}{c|cccc} & w1 & w2 & w3 & w4 \\ \hline w1 & 0 & 0.5 & 0 & 0.5 \\ w2 & 0.5 & 0 & 0.5 & 0 \\ w3 & 0.25 & 0.25 & 0.25 & 0.25 \\ w4 & 1 & 0 & 0 & 0 \end{array}$$

Finally, the values in this matrix will be updated in an iterative fashion to arrive at the web page rankings.

Text Rank Algorithm follows the similar fashion as the above Page Rank algorithm except it uses sentences in place of web pages and the similarity between any two sentences is equivalent to the web page transition probability. The flow of the Text Rank algorithm will be as follows:



1. The first step would be to concatenate all the text contained in the articles
2. Then split the text into individual sentences
3. In the next step, we will find vector representation (word embeddings) for each and every sentence
4. Similarities between sentence vectors are then calculated and stored in a matrix
5. The similarity matrix is then converted into a graph, with sentences as vertices and similarity scores as edges, for sentence rank calculation
6. Finally, a certain number of top-ranked sentences form the final summary

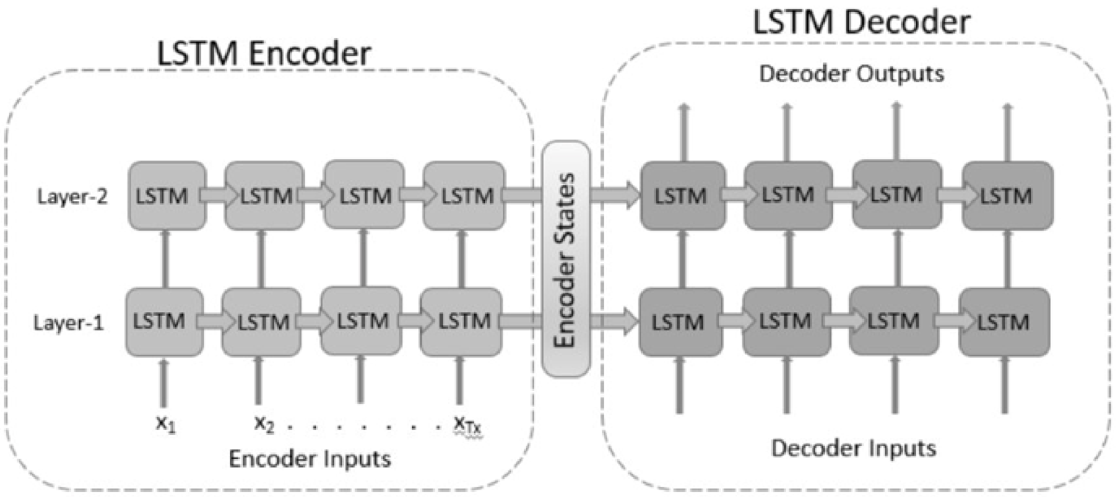
Seq2Seq (LSTM) Model

LSTM long short term memory is an artificial recurrent neural network(RNN) which is an approach for abstractive text summarization using the seq2seq model. In the seq2seq model, when given an input to the encoder decoder seq2seq model, it first generates an encoded representation of the model, which is then passed to the decoder to generate the desired output.

Abstractive summarizers are so-called because they do not select sentences from the originally given text passage to create the summary. Instead, they produce a paraphrasing of the main contents of the given text, using a vocabulary set different from the original document. This is very similar to what we as humans do, to summarize. We create a semantic representation of the document in our brains. We then pick words from our general vocabulary (the words we commonly use) that fit in the semantics, to create a short summary that represents all the points of the actual document.

We can set up the Encoder-Decoder in 2 phases:

1. Training phase
2. Inference phase

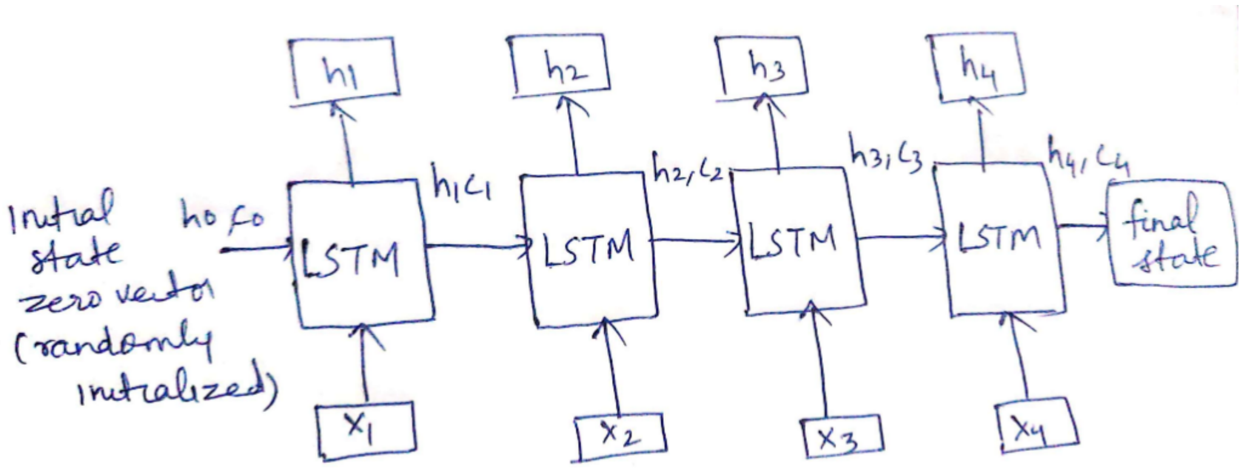


a) **Pre-processing :** Performing basic preprocessing steps is very important before we get to the model building part. Using messy and unclean text data is a potentially disastrous move. So in this step, we will drop all the unwanted symbols, characters, etc. from the text that do not affect the objective of our problem.It involves the conversion of all words in lowercase, expanding the contractions in the text, removal of punctuations, removal of special characters, and removal of stop words from the text. We will perform the below task as part of cleaning the data.

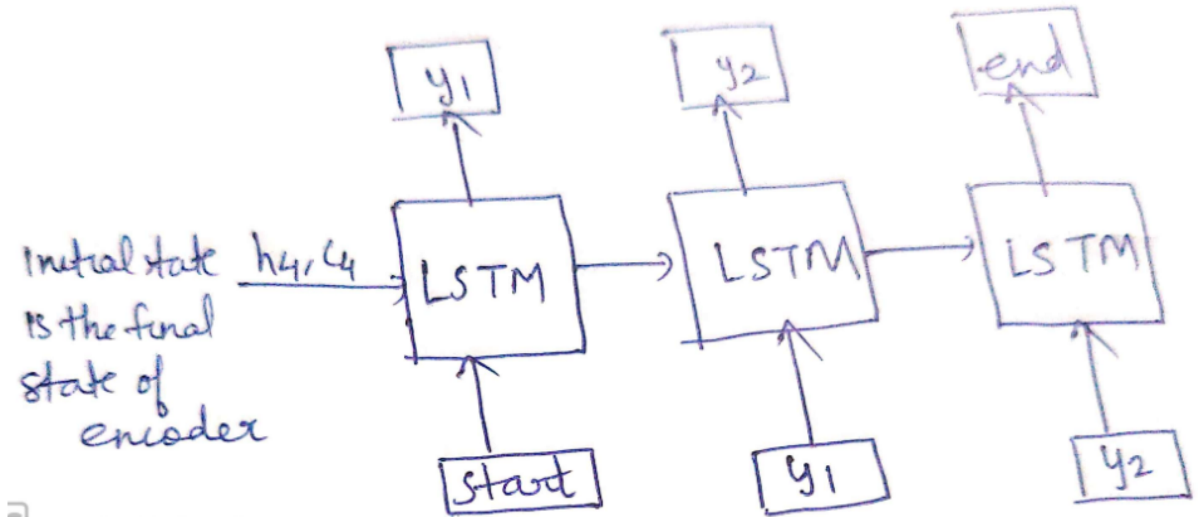
1.Convert everything to lowercase	2.Remove HTML tags
3.Contraction mapping	4.Remove (‘s)
5.Remove any text inside the parenthesis ( )	6.Remove stopwords
7. Remove short words.	8.Tokenization
9. Eliminate punctuations and special characters	

b) **Training phase**

Encoder : The encoder is made up of long short term memory units (LSTM). Among the variants of recurrent neural networks, LSTM overcomes the problem of vanishing gradients. Also, the LSTM’s are efficient for understanding the context of sequences that are very long. In order to process the text properly, the encoder must produce a context vector that captures the entire context. Hence use of LSTM is necessary concerning the context. Stacked LSTM are used in order to understand the sentences efficiently. Before encoding the sentence, the input text cannot be processed as such. It is converted into an integer format based on the term frequency in the sentence. Likewise, each sentence is converted into a sequence of integers based on their frequency and lookup dictionaries for each word along with their integers are built. Only after building the vocabulary, the sentence that is converted into a sequence is passed into the decoder. Below is the diagram for encoders.



Decoder :The input to the decoder is the previous state output of the decoder, the hidden state of the previous time step, and the output obtained through the encoder mechanism. The decoder calculates the target word probabilities based on these vectors at each time step. The greedy approach is followed for choosing the target word. Word with the highest probability is chosen and generated as the output of that time step. Likewise, an entire sequence is generated by the same mechanism.



c) **Inference Phase** :
In the inference phase, we want our decoder to predict our output sequence(which is summary). After training, the model is tested on new source sequences for which the target sequence is unknown. So, we set up the inference architecture to decode a test sequence.
But in the testing stage as mentioned earlier we do not know what the length of our target sequence would be. How do we tackle this problem? We follow the below steps for it :

1. Encode the entire input sequence and initialize the decoder with internal states of the encoder
2. Pass <start> token as an input to the decoder
3. Run the decoder for one time step with the internal states
4. The output will be the probability for the next word. The word with the maximum probability will be selected.
5. Pass the sampled word as an input to the decoder in the next time step and update the internal states with the current time step
6. Repeat steps 3–5 until we generate <end> token or hit the maximum length of the target sequence

Evaluation methodology :

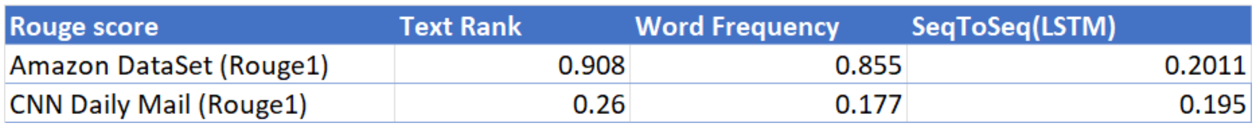
We have performed the evaluation of the summaries predicted using ROUGE metrics where ROUGE is an abbreviation of Recall Oriented Understudy for Gisting Evaluation is a metric used for the evaluation of automatic text summarization and machine translations. It basically compares the automatic generated summary with reference summary or multiple reference summaries. We have included two of the 5 evaluation metrics which are ROUGE-1 and ROUGE-L where ROUGE-1 measures overlap of unigram in reference summary and the predicted summary and ROUGE-L computes the longest common subsequence between the reference summary and the predicted summary.

Result Sample:

Amazon Review DataSet:
Review: husband gluten free food several years tried several different bread mixes first actually enjoys buying amazon saves loaf
Original summary: really good gluten free bread
Predicted summary: good bread

CNN Daily Mail DataSet:
Original Summary :
'Bishop John Folda, of North Dakota, is taking time off after being diagnosed .He contracted the infection through contaminated food in Italy .Church members in Fargo, Grand Forks and Jamestown could have been exposed .'
Predicted Summary :
' Associated Press . PUBLISHED: . 14:11 EST, 25 October 2013 . | . UPDATED: . 15:36 EST, 25 October 2013 . The state Health Department has issued an advisory of exposure for anyone who attended five churches and took communion. The diocese announced on Monday that Bishop John Folda is taking time off after being diagnosed with hepatitis A. Symptoms of hepatitis A include fever, tiredness, loss of appetite, nausea and abdominal discomfort.'

Result :



Rouge score	Text Rank	Word Frequency	SeqToSeq(LSTM)
Amazon DataSet (Rouge1)	0.908	0.855	0.2011
CNN Daily Mail (Rouge1)	0.26	0.177	0.195

Conclusion :

- We did the analysis of three algorithms text rank, word frequency, and seq2seq on amazon data set which is of small length reviews of 30 words, and CNN daily mails which comprise of medium size text comprises of 400-450 words to understand how the number of is making difference in each of the algorithms.
- For the amazon dataset word frequency performs the best as the reviews size of 30 words, however, text rank could not converge the results as it ranks the sentences and 30 words in a review is not enough sentences to be ranked so it gave the full sentences as the summary.
- Seq2Seq model we cannot train CNN daily mail dataset for 287000 data, therefore we had to cut down the data to 80000 and decrease the number of epochs. If we train the model on a high GPU machine for more epochs and with a full dataset then the rouge score will increase.

Future Experiment :In the SeqToSeq model we can use the attention layer to retrieve better-predicted summaries and we will work on incorporating the attention layer to increase the rouge score for abstractive text summarization.

Citations :

1.https://web.eecs.umich.edu/~mihalcea/papers/mihalcea.emnlp04.pdf
2.https://media.neliti.com/media/publications/166330-EN-single-document-automatic-text-summariza.pdf
3. Abstractive Text Summarization using Seq2seq Model (ijcaonline.org)
4. https://www.tensorflow.org/