CS 6120: Text Summarization using NLP

Sri Charan Vemuri, *Student - CS6120* **vemuri.sri@northeastern.edu**

Shri Datta Madhira, *Student - CS6120* **madhira.s@northeastern.edu**

Rahul Reddy Baddam *Student - CS6120* **baddam.ra@northeastern.edu**

Abstract

In the present era, as natural language processing (NLP) becomes increasingly important, text summarization has emerged as a valuable technique to quickly understand large volumes of textual data. In this project, we aim to explore two main types of text summarization: Extractive (ES) and Abstractive Summarization (AS) techniques using Amazon Fine Food Reviews dataset. We will evaluate the performance of both techniques using the BLEU and ROUGE scores. By doing so, we hope to develop a more comprehensive understanding of the pros and cons of each approach and offer insights into the most effective practices for summarizing text.

1 Introduction

Text summarization is a natural language processing technique that aims to automatically generate a shorter version of a given text document while retaining the most important information. The objective of text summarization is to help readers quickly understand the main ideas and key points of a long document, without having to read the entire document. There are two main types of text summarization methods.

1.1 Extractive Summarization (ES) is an approach in which the summary is generated by selecting and combining the most important sentences or phrases from the original text document. The selected sentences are usually taken verbatim from the original text without any modification. Some of the most popular methods for Extractive Summarization are, Latent Semantic Analysis (LSA), LexRank, TextRank, SumBasic, Luhn Summarization, Edmundson Summarization.

1.2 Abstractive Summarization (AS) is an approach in which the summary is generated by paraphrasing and rewording the original text to create a shorter version that captures the most important information.

Abstractive summarization requires the model to have a deeper understanding of the meaning of the text, as it often involves generating new text that is not present in the original document. A few methods of Abstractive Summarization are, Sequence-to-sequence models with LSTM, Sequence-to-sequence models with Attention, and Pointer-generator networks.

We will be using BLEU and ROUGE scores as metrics for evaluating the efficiency of our extractive and abstractive summarization models. The use of these metrics is essential in providing a quantitative measure of the quality of the generated summaries. The BLEU score, usually [0, 1], measures the similarity of machine produced text to high quality reference texts. ROUGE score is later developed taking inspiration from BLEU score for auto summarization tasks. In simpler words, BLEU score calculates precision, whereas ROUGE score calculates recall.

2 Background/Related Work

The field of text summarization has a long and rich history dating back to the early 1950s, when researchers first began exploring methods for automatically summarizing news articles. In the early years, most of the work focused on extractive summarization, which involves selecting and combining the most important phrases from the original text document to create a summary. This was done using various statistical and linguistic techniques, such as frequency analysis, sentence clustering, and semantic analysis.

In the late 1990s, researchers began exploring abstractive summarization techniques, which involve paraphrasing and rewording the original text to create a summary that captures the most important information. With the recent advancements in deep learning and natural language processing, abstractive summarization

has become a major area of research, and several state-of-the-art models have been proposed, including sequence-to-sequence models with LSTM, attention mechanism, and pointer-generator networks.

3 Data

We will not be annotating data ourselves and will be using readily available dataset. The Food Review Dataset is freely available and downloadable. The dataset used for this project is sourced from Kaggle. The dataset contains approximately **568,454** food reviews from Amazon covering a period of more than 10 years. This dataset contains a large collection of reviews for various food products sold on Amazon, along with other information such as the product ID, reviewer ID, review text, rating, helpfulness votes, and more. With over half a million reviews, this dataset provides a rich source of information for studying text summarization techniques.

We took 100000 reviews from the dataset. Data preprocessing and cleaning was performed on the dataset to convert raw text data into a cleaned version that contains only useful information. The cleaning process involved several stages including dropping duplicates and null value rows, removing numbers, punctuation, special characters, HTML tags, email addresses, URLs, and unnecessary white spaces. The text was also normalized by converting all characters to lowercase and removing accent characters. Contraction mapping dictionary is used to expand contractions in the text data.

Examination	Result
Check for null values in text and summary columns	0
Number of reviews	88421
Vocabulary size	66898
Check for duplicates	0
Average Document Length	81.5
Maximum Document Length	2520

Table 1. Analysis for Amazon Review Dataset after preprocessing 100,000 reviews.

We also used News Paper Dataset as a second dataset to see the performance of our models, but Food Reviews Dataset is the one which we will be exploring further in the Results section.

3.1 Exploratory Data Analysis

The below graph shows (Fig 1) the 20 most common words in the text data. The y-axis represents the words, and the x-axis represents the frequency of occurrence of each word. The word "the" having the highest frequency, followed by "I" and "and". The graph provides a quick and clear visual representation of the most common words in the text data before text cleaning.

This information can be used to gain insights into the most common combinations of words in the text data.

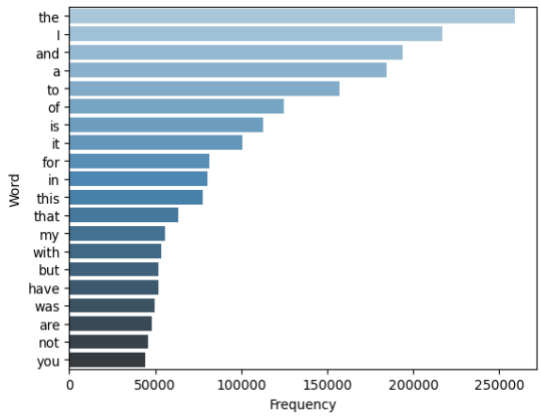

Fig 1. The most common words in the text data

The output of the graph below (Fig 3), shows a word cloud of the most common words in the text data. The larger the word in the cloud, the more frequently it appears in the text data.

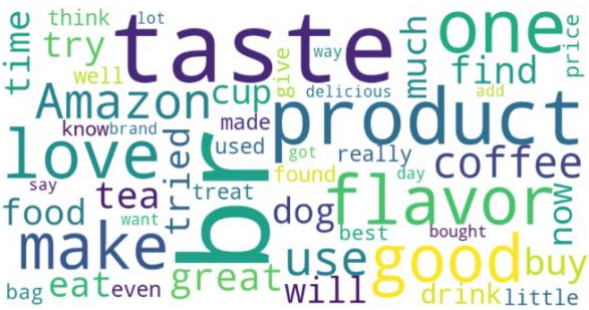

Fig 2. Word cloud of the most common words

4 Models

We will implement multiple Extractive Summarization models and discuss the results achieved on the dataset. We also implement two sequence-to-sequence (represented from now as Seq2Seq) Abstractive Summarization models that use Long Short-term Memory layers and Attention mechanism.

4.1 Extractive Text Summarization Models

Luhn Summarization is a heuristic method for extractive text summarization. The method is based on the idea that important sentences contain the most significant words. It uses a list of stop-words to identify significant words. The algorithm

calculates a score for each sentence based on the frequency of significant words it contains. Sentences with the highest scores form the summary. However, it can produce poor results without a proper list of stop-words for the language.

Latent Semantic Analysis or LSA is an algebraic method for text summarization that uses linear algebra to analyze relationships between words in a text. LSA is based on the idea that words that are used in similar contexts are related in meaning. This method creates a "semantic space" where each word is represented as a vector of numbers that capture its meaning. By comparing the vectors of different words, it can identify synonyms and related concepts in the text.

LexRank, is another extractive summarization method that is based on transforming a document into graph of sentences, where edges between the sentences are based on their semantic similarity. This method uses the PageRank algorithm to identify the most important sentences in a document, with higher scores indicating more important sentences. Selects the top-ranked sentences to form the summary.

TextRank, uses graphs for implementing extractive summarization. Like LexRank, this method uses a graph of sentences, where edges between sentences are based on their similarity. Ranks the sentences and selected the top-ranked sentences to form the summary.

4.2 Abstractive Text Summarization using Seq2Seq Modeling

Recurrent Neural Network is a neural net that contains a cycle within its network connections, meaning the value of some unit is directly, or indirectly, dependent on its own earlier outputs as inputs. A simple recurrent network (also called Elman Networks (Elman, 1990)) serves as a base for more complex approaches like LSTM.

RNNs for language modeling process the input one word at a time, attempting to predict the next word using the current word and hidden states. RNNs don't suffer from the limited context problem that n-gram language models suffer with. But these networks suffer from two issues, carrying the relevant information forward, and vanishing gradient problem.

Long Short-Term Memory Network is a complex network architecture based on RNN. It deals with the context management problem by dividing it into two sub-problems: forgetting information no longer useful, retaining information likely to be useful later in decision making. LSTM units are added as layers to our seq2seq models.

Sequence-to-Sequence (Seq2Seq) models are a popular choice for text summarization. Their architecture consists of two RNNs – an *Encoder* and a *Decoder*. The encoder processes the input text and creates a hidden representation, which is then sent fed into the decoder. The decoder generates the summary using the hidden representation. In this paper, we will be experimenting with two types of Seq2Seq models, Basic Seq2Seq models, and Attention-based Seq2Seq models. There are also Pointer-generator networks and Hierarchical seq2seq models.

Basic seq2seq models are the simplest versions of the model, where the input text is fed into the encoder that generates a hidden representation which is then turned into a summary using decoder. **Attention-based seq2seq models** use an attention mechanism that helps the decoder pay attention to specific parts of the input text while summarizing. This helps the model to focus on the most important information in the input text.

5 Experiments

5.1 Extractive Summarization Models

We have implemented ES models mentioned above using the sumy python package. The input text is first preprocessed by extracting and cleaning the text. The text is then parsed using a plain text parser and 2 sentence summaries were generated by passing the parsed text through multiple summarizer models. Stop word are ignored by the model when generating summaries. The candidate summaries are compared with reference summaries to generate BLEU and ROUGE scores. We have produced BLEU, ROUGE scores, candidate (machine generated) summaries generated by LSA, Luhn, TextRank, LexRank summarizers for 88,421 sentences.

5.2 Abstractive Summarization Models

5.2.1 Basic Seq2Seq Model using LSTM

A basic sequence-to-sequence (Seq2Seq) model that employs an encoder-decoder architecture using LSTM layers to generate summaries of input texts. The input text is first tokenized and then passed through an LSTM encoder layer to capture the context from both directions of the input sequence. The decoder LSTM layer takes the previous time step's output and the hidden state from the encoder as input to generate the summary word by word until the end token is reached. The model is trained using the RMSProp optimizer and categorical cross-entropy loss function, which is suitable for multiclass classification problems.

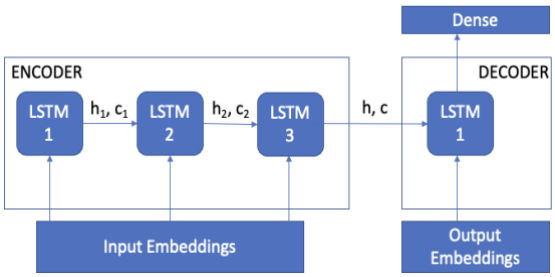

Fig 3. Model-1 flowchart

To handle out-of-vocabulary words, the Tokenizer is configured to either ignore or replace them. The model's hyperparameters are set according to the maximum and minimum length of texts and summaries. The embedding size is set to 200 and latent dimension is set to 300, with dropout and recurrent dropout values of 0.4 in the encoder and 0.2 in the decoder LSTM layers. These values may be adjusted during experimentation.

5.2.2 Attention-based Seq2Seq Model using LSTM

We add attention mechanism to our previous model to check the impact of this approach. The Attention mechanism is not provided directly in Keras packages, so we will be using a third-party implementation which can be found here. The third-party implementation is based on Bahdanu approach of global attention (paper in references). The model begins by passing the input sequence through LSTM encoder layers to generate a context vector. This vector is then fed into the decoder. The incorporation of the attention mechanism enables the decoder to focus on specific parts of the input sequence when generating the summary, thereby improving the generation of legible summaries that are close in meaning to original summaries provided.

Like 5.2.1, this model utilizes an approach to exclude rare and less coverage words from the vocabulary. The model is trained using the same optimizer and loss function as in 5.2.1. During decoding, a greedy search algorithm is used to generate the summary, where the token with the highest probability at each time step is chosen as the next token in the summary. Alternatively, beam search decoding can also be employed to generate multiple candidate summaries and select the best one based on a scoring function.

The attention mechanism used provides an improvement over Model 1, as it enables the model to selectively attend to the most relevant parts of the input sequence when generating each output token, thus enhancing the legibility of the generated summaries. However, it is important to note that a larger vocabulary may increase the training time and memory usage, thus, requiring careful balancing of the vocabulary size with the available resources.

Model	***Model 1***	***Model 2***
Architecture	Seq2Seq with LSTM	Seq2Seq with LSTM
Attention Mechanism	None	Yes
Encoder	Embedding Layer 3 LSTM layers	
Decoder	Embedding Layer LSTM Layer Dense Layer	
Vocabulary Handling	Excludes rare words and words with less coverage, Excludes sentences above a certain length	

Table 2. Abstractive Summarization models

After the models are trained, we need to set it up to generated predictions. For predicting the next word, the encoder feeds its output as input to the decoder in a word embedding format. The decoder then outputs a vector representations of the next token. We then take the argmax of these representations to map it to the next token.

6 Results

6.1 Extractive Summarization Models

Out of the four extractive summarization methods used, LexRank had the best average BLEU score of 3.18 and a 9.24 Rouge score. Luhn summarization technique is the second-best technique with an average BLEU 2.87 and 8.71 ROUGE.

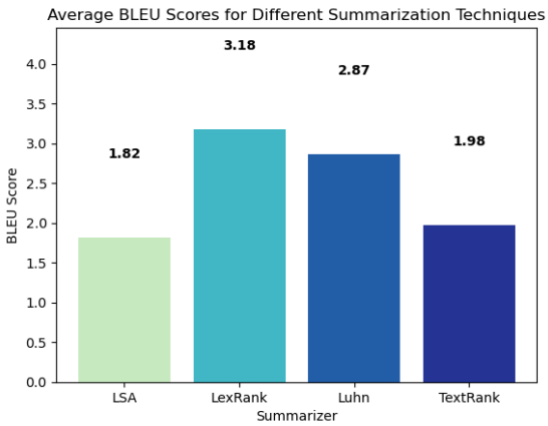

Fig 4. Extractive Models – BLEU scores.

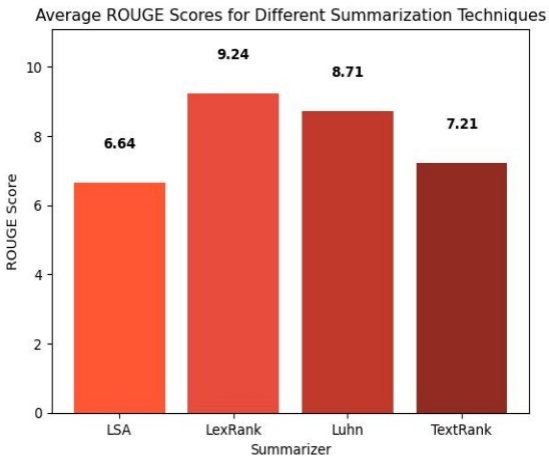

Fig 5. Extractive Models – ROUGE scores.

We believe the reason for LexRank to be at the top, is because it is based on the concept of eigenvector centrality, which means that it assigns higher importance to sentences that are connected to other important sentences in the document. This helps it to identify the most important sentences in the document. Similarly, Luhn's method of identifying important sentences is based on the frequency of important words in the sentence, which can help to identify sentences that are likely to contain important information.

6.2 Abstractive Summarization Models

The plots show the loss for train and test sets (i.e., training loss and validation loss). We use an early stopping mechanism that stops training the model by monitoring a user specified metric. In our models, we used validation loss as a metric to be monitored, means when the val_loss increases the model training is halted.

6.2.1 Basic Seq2Seq Model using LSTM

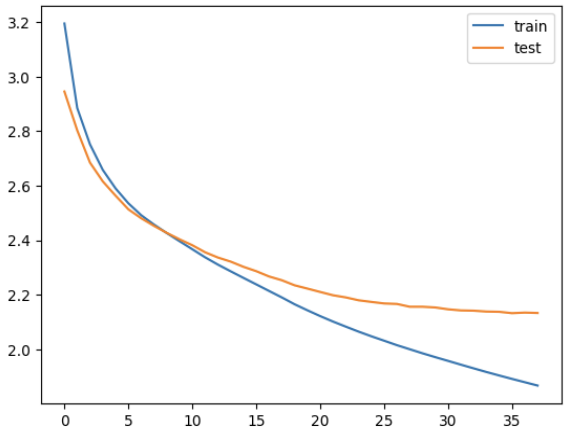

Fig 6. Model-1 train vs test set loss.

The model took approximately 3 hours to run on Google GPUs and took 38 epochs of batch size of 128 per epoch. We tried to add more LSTM layers, while we did see a slight improvement, we believe the improvement is too little to the extra time added for training.

The figure below shows some examples of predicted output summaries. As we can see, even though the predicted summaries are neither of the same length, nor contain the same words, the overall meaning is still conveyed.

```
Text: san francisco bay coffee especially good french roast body flavor french
roast usually get keurig reasonably priced would likely change brands
Original summary: san francisco bay coffee
Predicted summary:  good coffee


Text: love kashi products one favorites cereals like mix couple brands high
fiber low fat breakfast eat fine untill almost everyday especially good sliced
fresh strawberries
Original summary: healthy breakfast option
Predicted summary:  great cereal


Text: trust bad reviews right got pretzels favorite brand longer carried cost
plus remembered trader joe know keep buying taste stale really awful waste money
Original summary: stale and bad
Predicted summary:  stale


Text: long standing favorite classic hellman mayonnaise taken next level
addition mustard enhances flavor sandwiches salads squeeze cap standing bottle
makes much likely used takes little room refrigerator could without product even
like mustard would think making ham swiss croissant sandwich without
Original summary: delicious
Predicted summary:  great for cooking
```

Fig 7. Model-1 predicted summaries.

The BLEU and ROUGE scores are better than out top extractive models with BLUE score of 11.4 and ROUGE score of 15.8. These scores will improve with increase in training data, addition of more LSTM layers, and other improvements in the model.

6.2.2 Attention-based Seq2Seq Model using LSTM

This model took approximately 2 hours to run on Google GPUs and took 25 epochs with batch size of 128 per epoch. The plot looks like in Fig 6 which is expected because the only change is the addition of attention layer.

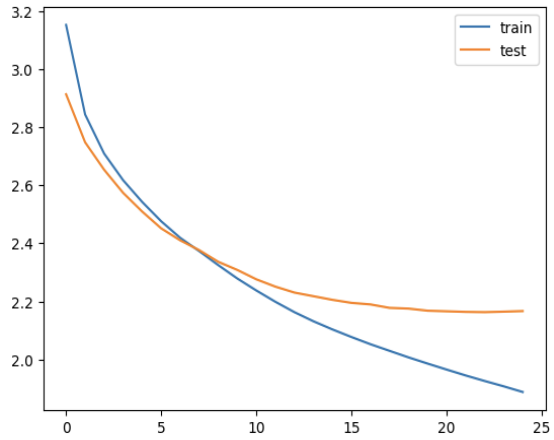

Fig 8. Model-2 train vs test set loss.

The predicted summaries are close to the original summaries, but we did not observe a significant improvement from the model without attention. We believe this is down to the small dataset size we took for training and a relatively simple model to accommodate for time and resource constraints. The BLEU score of the model is 11.6 and ROUGE score is 15.1. The scores are similar or a slight improvement from model-1 which we already inferred by looking at predicted summaries.

```
Text: cat became ill like another poster vomiting throughout day concerned also
diarrhea never happened repurchasing
Original summary: cat very
Predicted summary:  cat food


Text: great self feeding son getting picky still loves pouches plus great
company organic uses gmo ingredients
Original summary: great product
Predicted summary:  great food


Text: amazing great tasting low calorie carbs got lot packs great deal
Original summary: great
Predicted summary:  delicious
```

Fig 9. Model-2 predicted summaries.

7 Conclusions and Future Work

Extractive summarization models provided useful summaries but did not make any meaningful insights to generate different words or sentences to summarize text. There is no learning involved. Even though the actual and predicted summaries do not exactly match up, all the abstractive models did a good job of conveying the meaning from the text. We believe this to be a decent result with the small amount of training data and a simple model we took to explore this area of NLP.

Encoder-Decoder models have disadvantages of their own, so does Attention mechanisms. Researchers have found out that BLEU scores encoder-decoder models fall of drastically with the increase of text lengths (> 20 shows worse performance). We can incorporate more LSTM units or Bidirectional LSTM units, beam search technique, or pointer generator networks for improved performance.

Acknowledgments

I would like to express my gratitude to Professor Silvio Amir for his guidance and support in this project. His valuable insights and suggestions have been instrumental in shaping this project and taking it to completion. I would also like to extend my thanks to our Teaching Assistants, Pavan Sai Guduru, Sushant Kumar Jha, and Thean Cheat Lim, for their constant support and timely feedback.

References

https://arxiv.org/pdf/1409.0473.pdf

https://medium.com/@iit2018056/text-summarization-with-attention-based-networks-8492e9277c0

https://towardsdatascience.com/understanding-automatic-text-summarization-2-abstractive-methods-7099fa8656fe

https://towardsdatascience.com/text-summarization-using-deep-learning-6e379ed2e89c

https://towardsdatascience.com/text-summarization-from-scratch-using-encoder-decoder-network-with-attention-in-keras-5fa80d12710e

https://stackoverflow.com/questions/44924690/keras-the-difference-between-lstm-dropout-and-lstm-recurrent-dropout

https://arxiv.org/pdf/1706.03762.pdf

https://arxiv.org/abs/1601.06733

https://miso-belica.github.io/sumy/

https://towardsdatascience.com/understanding-automatic-text-summarization-1-extractive-methods-8eb512b21ecc

H. P. Luhn, "The Automatic Creation of Literature Abstracts," IBM Journal of Research and Development, Vol. 2, No. 2, 1958, pp. 159-165. http://dx.doi.org/10.1147/rd.22.0159