Team26: ProjF Final Report: Text Summarization using Deep Learning

Himanshu Gupta
hgupta6@ncsu.edu

Tirth Patel
tdpatel2@ncsu.edu

Harsh Kachhadia
hmkachha@ncsu.edu

I. Motivation and Problem Task

A. Motivation

The major motivation for this project comes from our day-to-day lives where we read the long paragraphs of text to get the main idea presented within it. There is enormous amount of text material available online as well as offline for any topic which is continuously growing day by day. A lot of times we prefer to skip reading the text after going through its length and so we miss getting some great content.

According to Radev et al. [3] a summary is defined as "a text that is produced from one or more texts, that conveys important information in the original text(s), and that is no longer than half of the original text(s) and usually, significantly less than that". Thus, the main aim for this project is to build a model which will be able to generate a brief summary of the provided input text.

B. Problem Task

There has been major development in Natural Language Processing and Deep Learning in the past few years which allows us to understand the language context and help us provide a brief summary about it. We aim to tackle the problem mentioned in motivation section by building a deep learning model using Long Short-term Memory (LSTM) and Sequence-to-Sequence modeling using Encoders and Decoders which can understand the context and provide a contextual summary of the input paragraph.

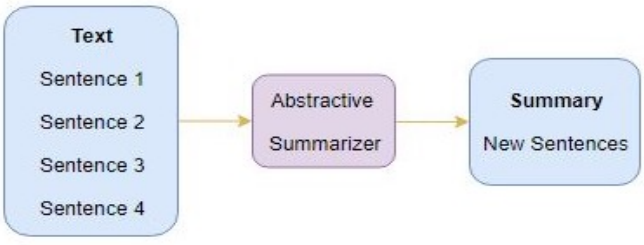

Fig. 1. Overview of Abstractive Text Summarization

II. Dataset and Preprocessing

A. Dataset

The dataset we are using in this project is "Amazon Fine Food Reviews Dataset (Version 2)" which is available in Kaggle through Stanford Network Analysis Project. The data include 568,454 samples that contain product and user information, ratings, long plain text review, and their brief summary.

B. Preprocessing

The input for our model will be the long paragraph of review text and the model will generate the brief contextual summary. So, other unnecessary information from the dataset was removed and only Review Text and Summary columns were kept which will be used for the current task at hand.

From our filtered data, we removed all the empty rows, duplicate reviews, and rows with NA, after which, basic text preprocessing steps were performed on reviews and summary text which consisted of cleaning the data such as removal of punctuation, stop words, special characters, conversion to lower case, etc.

Later, from the analysis of review text and summaries, 94% of summaries and reviews were found to have had the length of 8 and 30 respectively, thus, maximum sequence length for reviews and summaries were set to 30 and 8 respectively. Rest of the reviews and summaries which didn't meet these length requirement were dropped.

The textual data needs to be converted into series of integers or floating point numbers so that it can be fed as input to the machine learning/deep learning models. This process of conversion of textual data to vectors are called vectorization. We used Tokenizer() from TensorFlow. Keras library which provides many more features such as word_count, word_index. Tokenizers for both reviews(both training and test) and summaries(only training) were built which build the vocabulary and convert a word sequence to an integer sequence. Additionally, padding using the 'post' method was used, in order to make the sequences of uniform length.

We removed rare words (words with frequency less than threshold) by setting word count threshold to 6 for

summaries(only training) and threshold to 4 for reviews(both training and test).At the end, we removed empty review or summary sequences present with only start and end tokens. This preprocessing of dataset makes it ready as per the expectation for input to our NLP model.

III. Methodology

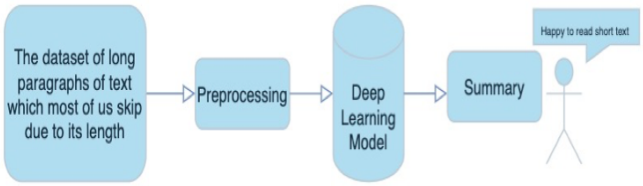

Fig. 2. Overview of Text Summarizing Process

There are 2 different types of summarization approaches

1) Extractive Summarization
2) Abstractive Summarization

Extractive Summarization : In this approach, the model identifies the most important sentences from the paragraph and extract only those sentences which will be treated as the summary for the paragraph. Figure 3 shows the overview of Extractive Summarization.

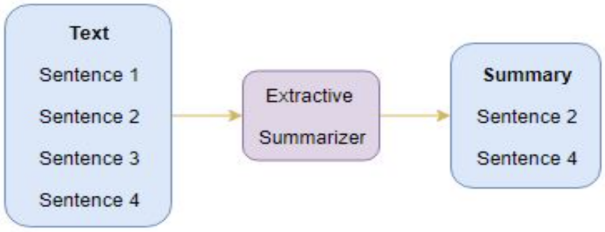

Fig. 3. Overview of Extractive Summarization

Abstractive Summarization : In this approach, the model tries to generate new sentences from the paragraph which will summarize the content from the paragraph. Refer to Figure 1 for the overview of Abstractive Summarization.
For our project, we choose to implement a deep neural network to perform Abstractive Summarization.

We plan to use Many-to-Many Seq2Seq model for text summarizing. A Seq2Seq model is used at places which requires sequential information. Some common applications of Seq2Seq models include name entity translation, sentiment classification, neural machine translation, etc. Our target here is to build a text summarisation model which will take long sequences of words as input and produce a short summary as output.

A Seq2Seq model consists of two parts, namely Encoder and Decoder. Usually some kind of variant of Recurrent Neural Network (like Gated Recurrent Neural Network (GRU) or Long Short Term Memory (LSTM)) are used for the encoder-decoder architecture. For this project we plan to use Long Short Term Memory(LSTM) as encoder and decoder component. LSTM is an artifical Recurrent Neural Network architecture which uses the feedback connection to process large sequences of data.

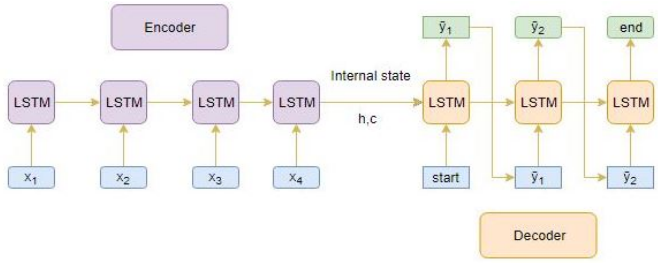

Fig. 4. Encoder-Decoder LSTM architecture

An encoder LSTM reads the entire input sequence and at each step, one word is fed into the encoder. Encoder LSTM will try to gain the context of the input information. On the other hand, decoder is used to predict the next word when some previous word is given. Two special tokens, namely, <start> and <end> are used to help in start predicting the output sequence and end the output sequence respectively.
Once the model has been trained, we need to test the model to see how well it performs. To do so, we will first encode the input and initialize decoder using the states from the encoder. We then pass <start> token into the decoder as the input. The output of decoder will be the next word in the sequence(the word with maximum probability is selected). The output word is then passed as input to the decoder which has updated parameters from the previous state. Repeat these steps till we encounter <end> token.

A. Model Training and Selection

We divided our dataset into 90% training and 10% test dataset. We will be using this 10% dataset for the evaluation of our model.
To perform the summarization task at hand, we chose to implement LSTM and Bidirectional-LSTM models. We used Adam optimizer with learning rate=0.01 for our models. Figure 5 shows the training and testing loss curve for 1-layer LSTM model.

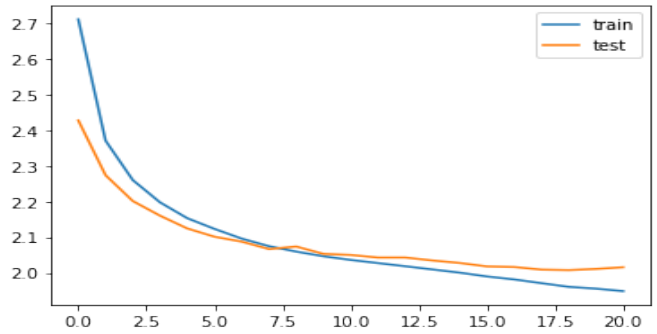

Fig. 5. 1-Layer LSTM graph

We also performed **Hyper-parameter tuning by increasing the number of layers in LSTM and varying the learning rate.** Figure 6 shows the training and testing loss curve for 3-layer LSTM model.

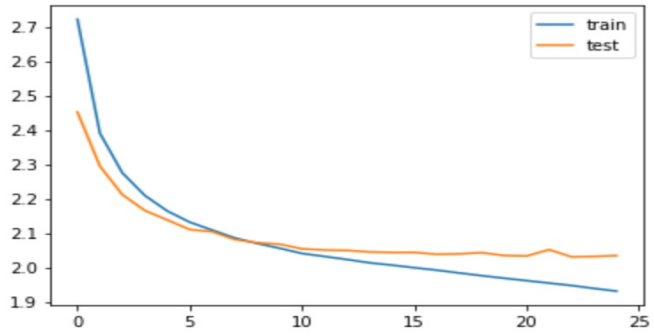

Fig. 6. 3-Layer LSTM graph

For our final model, we chose to implement Bidirectional LSTM model because of it's capability to capture the context from both the directions because the data is fed into the model from both the sides. Figure 7 shows the Bidirectional LSTM architecture explaning how the data is fed into the model from both the directions.

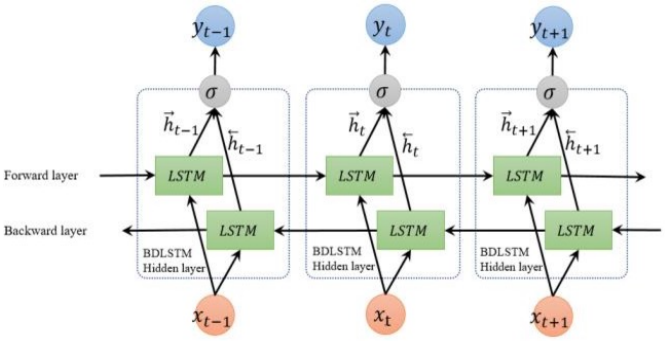

Fig. 7. Bidirectional LSTM architecture

Bidirectional model outperforms the previous model and was able to generate more accurate summary for the data given. Figure 8 shows the training and testing loss curve for 3-layer LSTM model.

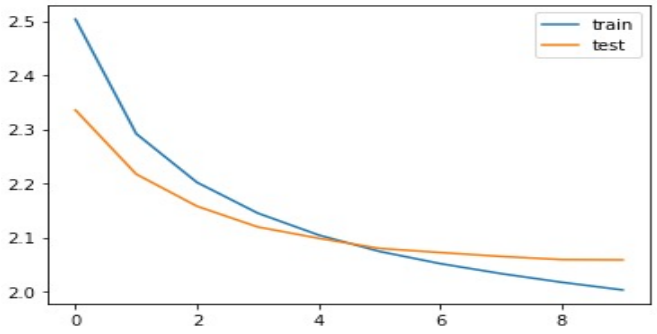

Fig. 8. Bidirectional LSTM architecture

IV. Evaluation and Results

As stated earlier, we divided our dataset into 90% training and 10% test dataset. We will evaluate the model based on the accuracy of the summary generated for the 10% test dataset. We will also evaluate our same input sequence against SpaCy text summarization library which will serve as our baseline model.

Human evaluation will also be required to check the contextual accuracy of the summary generated as the generated summary might not match to the expected summary word-by-word. If the contextual accuracy of generated summary match with the predicted summary, we will treat that instance as a "Success".

To evaluate the generated output summary of our model (including the baseline model, spaCY), we have used **Rouge Metric** which calculates the precision, f1-score and recall by comparing the original summary and the generated summary. Figure 9 shows these metrics generated through rogue for the test dataset.

	F1-score	Precision	Recall
1-Layer LSTM	0.13	0.41	0.07
3-Layer LSTM	0.17	0.71	0.10
Bidirectional LSTM	0.36	0.77	0.23
spaCY (Baseline)	0.18	0.69	0.12

Fig. 9. Evaluation Metrics for test samples

V. Test-case illustration

For illustration, we are taking a random review from our dataset. We then use that review to generate the summary through all our models.

- **Review** – one of the best choices so far. If you like chocolate coffee and good aroma you will like this one Will definitely buy again
- **Summary (Single Layer LSTM)** – good coffee

- **Summary (3 – Layer LSTM)** – great coffee buy again
- Summary **(Bi-directional LSTM)** – best chocolate coffee good aroma buy again
- **Summary (spaCY - (Baseline))** – great coffee and aroma

Figure 10 shows the evaluation metrics generated through Rogue for the above review and the summary generated through each model.

	F1-score	Precision	Recall
1-Layer LSTM	0.16	0.5	0.09
3-Layer LSTM	0.22	0.75	0.13
Bidirectional LSTM	0.47	0.82	0.3
spaCY (Baseline)	0.21	0.75	0.14

Fig. 10. Evaluation Metrics

As seen from the summary generated and the evaluation metrics, Bidirectional LSTM generates the most accurate contextual summary and performs better than all the other models.

REFERENCES

[1] Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi, Saeid Safaei, Elizabeth D. Trippe, Juan B. Gutierrez, Krys Kochut "Text Summarization Techniques: A Brief Survey" Cornell 2017.

[2] N. S. Shirwandkar and S. Kulkarni, "Extractive Text Summarization Using Deep Learning," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA).

[3] Dragomir R Radev, Eduard Hovy, and Kathleen McKeown. 2002. Introduction to the special issue on summarization. Computational linguistics 28, 4 (2002), 399–408.

[4] Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, Bing Xiang "Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond" Conll 2016.

[5] Dima Suleiman, Arafat Awajan, "Deep Learning Based Abstractive Text Summarization: Approaches, Datasets, Evaluation Measures, and Challenges", Mathematical Problems in Engineering, vol. 2020, Article ID 9365340, 29 pages, 2020. https://doi.org/10.1155/2020/9365340