Chaithra Kopparam Cheluvaiah, Megha Banerjee, Prateek kumar kumbar, reetodeep hazra

School of Information Studies

Syracuse University

**News Summarization**

Generating Abstractive summaries of news articles

May 9, 2022

**News Summarization**

**Chaithra Kopparam Cheluvaiah, Megha Banerjee, Prateek Kumar Kumbar,**

**Reetodeep Hazra**

**Keywords:**

LSTM, Text Summary, News Summary, T5 Transformer

**Introduction:**

Text summarization is an example of NLP that will certainly have a huge impact on our lives. With the quantitative rise of digital media and the proliferation of publishing, there is a business war going on between the media houses to capture the reader’s mind in every possible way. In today’s fast paced world, it has become almost impossible for people on-the-go to read every news on the newspaper (both paper and digital copy) to be up to date with the outside world. In this project, we have focused on the same business problem and approached the same in a data driven way.

Automatic Text Summarization (ATS) is a way to extract a short and coherent summary of text from a variety of sources, blogs, tweets, news stories, including books, emails, and research papers.

Two approaches to summarizing text:

Diagram, chat or text message

Description automatically generated

Figure 1: Overview of Text Summarization

* **Abstractive Summarization**

Abstractive Summarization produces more meaningful human written sentences, rather than being limited to terms from the source text.

Diagram

Description automatically generated

Figure 2: Abstractive Summary generation

* **Extractive Summarization**

Extractive summarization is a traditional way to generate summaries, such as clipping relevant chunks of the source text and combining to make a rational summary.

Diagram

Description automatically generated

Figure 3: Extractive Summary generation

**Literature Survey:**

A significant amount of work is done on text and news summarization focusing on generating extractive as well as abstractive summaries. Alexander et al. [4] used different data-driven approaches to perform sentence summarization. They performed Neural Language model and encoders to generate summaries based on the inputs and performed Bean-search algorithm for the feed forward model. Rush et al [5] used a sequence of input words and then mapped it in the form of a contextual probability distribution of previous outputs. They also created a condensed matrix using a decoder to maximize the contextual probability.

Coming to the application of machine learning in news summarization, Hujia et al [6] worked with attention based deep neural networks to build an automatic text summarization model for news articles. They reported an encoder-decoder model with LSTM as well as RNN and measured the respective ROGUE scores with actual references. The best ROGUE score came from a 3 and 4-layer RNN which is 20.52 and 18.77 respectively. Another work by Touseef et al [7] used Gated Neural Networks to review text summarization. The paper also used RNN as well as LSTM to summarize texts. In comparison with our work, Shengli et al [8] used LSTM-CNN based deep learning to perform abstractive text summarization. The authors constructed new sentences and semantic phrases from source sentences and used deep learning to generate text summaries. Abhijeet et al [9] used LSTM based encoder-decoder classification for extractive text summarization. The authors used three metrics to evaluate the model namely gold standard, ROGUE 1 and ROGUE 2. The model reported a ROGUE 1 score of 0.825 and ROGUE 2 score of 0.815 which is decent with respect to text summarization.

In this article, we have focused on generating abstractive summaries from newspaper articles using LSTM (Long Short-Term Memory) and T5-Text-to-Text Transfer transformer. The report is structured as following- Data set description, model implementation, generated summaries, followed by the conclusion.

**Data Set Description:**

For this project, we have used Indian news Dataset named News Summary [10]. It consisted of two individual datasets named ‘news\_summary’ and ‘news\_summary\_more’, with 4515 and 98280 instances respectively. Due to computational limitations, we have used only ‘news\_summary’ data set for this project. Here, out of 6 variables (author, date, headline, read\_more, text, and ctext), we have used only text (summary of the article) and ctext (complete article).

**Implementation:**

**Data Pre-processing:**

In every machine learning endeavor, data cleaning or preprocessing is just as important as model creation. Text data is one of the most unstructured sorts of data and dealing with human language is impossible. NLP is a technique that operates behind the scenes and performs extensive text preprocessing prior to any answer. The various text pre-processing steps are:

* **Handling null values:** Removing all the rows containing null values for news articles.
* **Lowercase Transformation:** Lowering the case of a word (NLP -> nlp).
* **Removing HTML tags:** Cleaning HTML tags before vectorizing the text data.
* **Stop words and punctuation removal:** Stop words (an, the, a, and so on) are frequently employed in papers. These terminologies have no real meaning because they do not assist in distinguishing between two publications.
* **Tokenization:** Tokenization is essentially splitting a sentence, paragraph, phrase, or an entire text into smaller parts, such as individual terms or words.
* **Lemmatization:** Unlike stemming, lemmatization restricts words to words that already exist in the language.

As an additional pre-processing step required for modeling seq2seq data, <start> and <end> special tokens are appended at the end and the beginning of the summaries.

**Exploratory Data Analysis:**

EDA stands for exploratory data analysis, which entails using summary statistics and graphical representations to identify patterns, spot abnormalities, test hypotheses, and confirm assumptions. As a part of EDA, we have performed the following data analysis.

* **Type token ratio (TTR):** The TTR is calculated by dividing the total number of different words (types) in a text by the tokens i.e. the total number of words. A high TTR suggests a lot of lexical diversity, whereas a low TTR shows the contrary. In our case, we got a 0.04 TTR score! As expected for news articles.

* **Word cloud:** Word cloud, which are also known as tag-clouds, are visual illustrations of word frequency that highlight words that appear frequently in a source text. The longer the term in the image, the more times it occurs in the text document (s).

A picture containing text, newspaper

Description automatically generated

Figure 4: Word Cloud

* **Top 20 bigrams:** A bigram would be a two-word sequence such as “I like”, “like NLP”, or” NLP course”. The picture below represents the top 20 bigrams in our corpus.

Chart, funnel chart

Description automatically generated

Figure 5: Top 20 bigrams

* **Top 20 trigrams:** A Trigram would be a three-word sequence of words like “I like NLP”, “like NLP course”. The picture below represents the top 20 trigrams in our corpus.

Chart, funnel chart

Description automatically generated

Figure 6: Top 20 trigrams

* **Top 20 frequent words:** Frequently occurring words are the words that appear highest number of times across the entire corpus. This kind of analysis help us in understanding the context of the data. The word ‘said’ came as the most frequent word found for 10570 instances. Other frequent words found in our corpus were ‘one’, ‘would’, ‘people’, ‘also’, ‘new’. Also looked at the hapaxes. Some results are ‘notables’, dupe’, ‘patronize’ and others.

Chart, line chart

Description automatically generated

Figure 7: Top 20 tokens by frequency

**Architecture:**

**Encoder-Decoder:**

Sequence to Sequence (seq2seq) models are a type of Recurrent Neural Network architecture that is commonly used to handle complicated language problems such as question answering, constructing chatbots, machine translation, text summarization, and so on. A Seq2Seq model has two primary components:

* **Encoder:** Encoder LSTM model scans the complete input sequence, with a word sent into the encoder at every time-step. It then examines the data at each timestep, capturing the contextual data in the input pattern. The cell state and last time step's hidden state (hi) are utilized to initialize the decoder.

**Diagram

Description automatically generated**

Figure 8: Encoder Architecture

* **Decoder:** The decoder is likewise an LSTM network that examines the whole target sequence word by word and predicts a one-step delayed sequence. It is taught to predict the following word in the sequence based on the preceding word. The special tokens start, and finish are added to the target sequence before it is fed into the decoder. The target sequence was unknown during decoding the test sequence. As a result, we start by feeding the decoder with the first word, which is always the <start> token. The <end> token denotes the conclusion of the sentence.

Diagram

Description automatically generated

Figure 9: Decoder Architecture

**Model Building and Execution:**

The model we built for the news summarization involved similar architecture of encoder and decoder as shown in the below image. Embedding dimension of 300 was used along with 3 layers of LSTM in the encoder and 1 layer of LSTM in decoder with a dropout 0.4. In the dense layer, softmax was used as an activation function.

Diagram

Description automatically generated

Figure 10: LSTM Architecture

**Model Summary:**

The model summary of the baseline model is as shown below. Encoder of the model contained 3 layers of LSTM and 1 layer of LSTM in the decoder part along with the input and embedding layers.

Table

Description automatically generated

Figure 11: Model Summary

**Model Evaluation:**

Our initial model results without hyper parameter tuning. The train loss and validation loss are plotted below. The validation loss is slightly higher than the train loss with an overall accuracy of 46% which is below the average model accuracy.

Chart, line chart

Description automatically generated

Figure 12: Training and Validation loss

**Limitations of Encoder-Decoder Architecture:**

This encoder-decoder design is quite useful, although it does have some drawbacks.

* The encoder predicts the output sequence after converting the full input sequence into a fixed size vector. Because the decoder looks at the full input sequence for the prediction, this only works for short sequences.
* The concern with long sequences is long sequences are difficult for the encoder to remember into a fixed length vector. For LSTM to perform better, according to BLEU score report, the length of input text should be less than 30 and the length of reference should be around 15.

Chart

Description automatically generated

Figure 13: BLEU Score vs. Sentence Length

**T5: Text-To-Text Transfer Transformer:**

The T5 transformer was presented by Google [2] in 2020 with the below research article.

Table

Description automatically generated with medium confidence

Figure 14: Limits of Text-to-Text Transformer; (source: GoogleAI)

The model structure is a common type of encoder-decoder transformer.

Diagram

Description automatically generated

Figure 15: Encoder Decoder Transformer

The T5 model is pre-trained on the Colossal Clean Crawled Corpus dataset which is called C4 dataset, and it achieves state-of-the-art results on a range of NLP tasks while still being flexible enough to be fine-tuned to a variety of tasks that we want to solve.

Diagram

Description automatically generated

Figure 16: T5 pre-trained model

Each task feeds text into a model that has been trained to create some target text. This enables the use of the same model, loss function, and hyperparameters for a variety of tasks as shown above - machine translation (highlighted in green), linguistic acceptability (highlighted in red), phrase similarity (highlighted in yellow), summarization (highlighted in blue).

**Fine Tuning T5 transformer for News Summarization:**

The HuggingFace Transformers hub has a T5 pre-trained model. For this project, we utilized the HuggingFace T5-base model. T5Tokenizer and T5-base model are included in the HuggingFace NLP library.

Prior to the actual article, a new string is added to the main article column 'summarize:'.  This is because the summary dataset in T5 was formatted similarly. On the console, the first 5 rows of the data frame are printed.

Text

Description automatically generated

Figure 17: Quick glance at the data

For test and validation, the updated data frame is partitioned in an 80:20 ratio.

A picture containing text

Description automatically generated

Figure 18: Train-test division

* To define our model, we utilized the T5ForConditionalGeneration.from pretrained("t5-base") command.
* To define the tokenizer, we used the T5 Tokenizer from the pretrained("t5-base") command.
* The Adam optimizer is being used.
* We limited the input target to have 512 tokens and 150 for the output summary.

**T5 Base Model Evaluation:**

With a batch size of two, we trained the model for two epochs. Every 500th step, the training loss is displayed on the console.

Text

Description automatically generated with medium confidence

Figure 19: Base Model Evaluation

Training loss was quite high in the first iteration. However, it significantly reduced in the later iterations. We have used ROUGE metrics to evaluate the performance of the model.  It is commonly used for evaluating text summarization as well as machine translation. ROUGE works by comparing model-generated summary with labeled summary.

Text

Description automatically generated

Figure 20: Rouge matrix

The count of unigrams shared between the generated and reference summary is known as ROUGE-1. Between the generated and reference summaries, 46% of the tokens are overlapping on an average.

ROUGE-2 refers to the count of bigrams shared between generated and reference summaries. On an average 28% of the bigrams are overlapping.

ROUGE- L determines the longest matching subsequence of words. The length of the largest sequence of tokens shared between both summaries is 38.64% on average.

**Summary Generated from the T5 Transformer:**

**Predicted Summary - 1**

**Text, letter

Description automatically generated**

**Predicted Summary - 2**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Text

Description automatically generated**

**Predicted Summary - 3**

**A close-up of a document

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**Conclusion:**

Extractive Text Summarization models focus on syntactic structure, whereas Abstractive Text Summarization models focus on semantics. The strengths of both summarization models should be included in our model. In this project, the LSTM and T5 Transformer model was used to compare each other’s performance and accuracy. According to the implementation results, T5 Transformer had better performance than LSTM under this circumstance. At least, the T5 Transformer model generated more readable and meaningful results based on these articles. In the future, we still need to test both models under different environments to see if they have better results.

**Acknowledgement:**

We sincerely thank Dr. Joshua Introne for his valuable suggestions and passionate support throughout the execution period of this project. Also, we would like to show our heartfelt gratitude to School of Information Studies, Syracuse University for providing us with the utmost technological support and other academic resources which helped us to complete the work within stipulated timeframe and high satisfactory.

**References**

[1] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., … Liu, P. J. (2019). *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*. doi:10.48550/ARXIV.1910.10683

[2] Exploring Transfer Learning with T5: the Text-to-Text Transfer Transformer, Google AI Blog: https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html.

[3] The Illustrated Transformer: Jay Alammar, Github: https://jalammar.github.io/illustrated-transformer/

[4] A. M. Rush, S. Chopra and J. Weston, “A Neural Attention Model for Abstractive Sentence Summarization”, 2015.

[5] Rush, A.M., Chopra, S. & Weston, J. (2015) A Neural Attention Model for Abstractive Sentence Summarization. 2015 Conference on Empirical Methods on Natural Language Processing (EMNLP).

[6] Yu, Hujia, Chang Yue and Chao Wang, “News Article Summarization with Attention-based Deep Recurrent Neural Networks” Stanford Natural Language Processing Group, Stanford University, pp.2746634, 2016.

[7] Iqbal, Touseef & Sambyal, Abhishek & Padha, Devanand. (2018), “A Review of Text Summarization using Gated Neural Networks”, INTERNATIONAL JOURNAL OF COMPUTER SCIENCES AND ENGINEERING, vol,6, pp. 5.

[8] Song, Shengli & Huang, Haitao & Ruan, Tongxiao. (2019). Abstractive text summarization using LSTM-CNN based deep learning. Multimedia Tools and Applications. 78. 10.1007/s11042-018-5749-3.

[9] Abhijeet Ramesh Thakare and Preeti Voditel, “Extractive Text Summarization Using LSTM-Based Encoder-Decoder Classification”, 2022 *ECS Trans.* Vol.107, pp.11665.

[10] News Summary, Kaggle: https://www.kaggle.com/datasets/sunnysai12345/news-summary