Automatic Text Summarization Techniques

Faranak Abri, Atharva Khadilkar, Monica Meduri, Mrunal Zambre
Computer Science Department, San Jose State University
{*faranak.abri, atharvapramod.khadilkar, monica.meduri, mrunaldeepak.zambre*}*@sjsu.edu*

***Abstract*—In today's age of big data, the amount of textual data from various sources has skyrocketed. This volume of text has a plethora of information and knowledge that must be adequately summarized in order to be useful. Summarization is the process of condensing a chunk of text into a shortened version, minimizing the size of the original text while keeping vital informational aspects and content significance. Manual text summarizing is a time-consuming and laborious activity, so automating the task is gaining popularity. Automatic text summarization can extract the key ideas from a large body of data, saving the time and resources of users, as well as facilitating quick access to information that needs to be searched within the documents. In this project, we perform text summarization on CNN / DailyMail dataset using TextRank Algorithm, T5, BERT-2-BERT (Fine-tuned on a subset of the dataset), and BERT-2-BERT(Fine-Tuned on the entire dataset).**

***Index Terms*—Automatic text summarization, extractive, abstractive, BERT2BERT, T5, neural networks, TextRank algorithm**

[1]

I. Introduction

Text summarization is the task of condensing the essence of a document to converse the main information contents and main ideas quickly. There are two paradigms to generate a summary: extractive summarization and abstractive summarization. Extractive summarization generates the summary by scoring and extracting the high-score sentences from source documents. On the other hand, Abstractive summarization generates a summary in a human-written way. In this project, we have implemented automatic text summarization using both the above methods. Extractive summarization was done using the TextRank algorithm. Abstractive summarization was done using T5 and BERT2BERT models.

II. Related Works

Since the dataset is curated and mostly cleaned, the CNN/dailymail news dataset is widely used for text summarization. The dataset was introduced by Nallapati et al. in 'Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond' [1] in which they attempted abstractive text summarization. The Rouge-F1 scores for the models proposed by Nallapati et al. 33.71%. This dataset has been used as a benchmark for multiple text summarization models such as T5 [2], Pegasus [3] etc. The highest rouge F1 scores achieved for this dataset are 43.87% using Pegasus+Summaranker by Mathieu Ravaut et. al. in 'SummaReranker: A Multi-Task Mixture-of-Experts Re-ranking Framework for Abstractive Summarization' [4].

[1]https://www.overleaf.com/read/vbykswytdptg

978-1-7281-6251-5/20/$31.00 ©2020 IEEE

III. Dataset Description

The CNN / DailyMail Dataset is an English-language dataset containing just over 300k unique news articles written by journalists at CNN and the Daily Mail. The current version supports both extractive and abstractive summarization. The data fields of this dataset are id, article, and highlights. This dataset has 3 splits: train, validation, and test containing roughly 280k,13k, and 11k samples respectively.

IV. Preprocessing

A. Handling missing data

The CNN / DailyMail dataset does not have any missing values.

B. Text Pre-processing for TextRank

For the TextRank algorithm, pre-processing of the text was required.
First, the entire text was tokenized into separate sentences. We used NLTK library's 'sent_tokenize' function for this. Then, for each sentence, all the digits and symbols were removed so that only letters remained. Further, all letters were converted into lowercase letters.
After this, all the stopwords in each sentence were removed. Stopwords are typically useless words and do not add much meaning to a sentence. In the English language common stopwords include "you, he, she, in, a, has, are, etc."

C. Text Pre-processing for T5

The pre-processing required for T5 was different than that required for TextRank. This is because T5 requires the input data to be in a very specific format. Our dataset originally had 3 columns, namely 'id', 'article', and 'highlights'. We had to remove the 'id' column and rename the other 2 columns to 'source_text' and 'target_text'. This was the expected data format for the input data for training the T5 model. To make predictions using the finetuned model, we had to add the prefix 'summarize: ' to the input string.

D. Text Pre-processing for BERT2BERT

Since the BERT model is not designed for text generation, we need to do some preprocessing. Our dataset originally had 3 columns, namely 'id', 'article' and 'highlights'. We remove the 'id' column and set up the tokenizer, and specify the beginning-of-the-sentence and end-of-the-sentence tokens to

guide training and inference processes correctly. It is defined in the model's config and its tokenizer object.

V. Summarization Algorithms and Models

A. *TextRank Algorithm*

TextRank is a graph-based ranking model for text processing that finds the most relevant sentences in a text. It is an extractive and unsupervised text summarization technique. It is similar to Google's PageRank algorithm.
We implemented the TextRank algorithm as follows:
1. The first step was to concatenate all the text contained in the articles.
2. Then, the text was split into individual sentences and pre-processed as explained in the above section.
3. In the next step, we found vector representation (word embeddings) for each and every sentence. This was done using gensim library's Word2Vec() function, which calculates embedding for each word of the text.
4. Then, similarities between sentence vectors were calculated and stored in a similarity matrix. We used cosine similarity to calculate the similarities between vectors.
5. The similarity matrix was then converted into a graph, with sentences as vertices and similarity scores as edges, in order to calculate the sentence ranks.
6. Finally, a certain number of top-ranked sentences were used to form the final summary. We chose the top 3 sentences as the summary, but any number could have been used here.
In this way, using the TextRank algorithm produces summaries from within the original text itself.

B. *T5*

Text-To-Text Transfer Transformer(T5) is a new transformer model from Google that is trained in an end-to-end manner with original text as input and modified text as output. Using a text-to-text transformer trained on a huge text corpus, it provides cutting-edge performance on a variety of NLP tasks such as summarization, question answering, machine translation, and so on.
T5 is an abstractive summarization algorithm. That means that it will rewrite the sentences whenever necessary rather than just picking up sentences directly from the original text. T5 is pre-trained on the Colossal Clean Crawled Corpus (C4), a cleaned version of Common Crawl that is two orders of magnitude larger than Wikipedia. Using this dataset led to better results by T5 on downstream tasks, while the additional size allowed the model size to increase without overfitting during pre-training.
We used the SimpleT5 library to fine-tune T5 model on our dataset as it provides an easier way to implement the fine-tuning.

C. *BERT2BERT*

BERT is a well-known and powerful pre-trained "encoder" model. We can use it as a "decoder" to form an encoder-decoder architecture. The encoder-decoder model, with the ability to condition the decoder's output based on the encoder's representation, can be used for tasks such as summarization and translation. It is done by having a cross-attention connection from the encoder to the decoder. Bert2BERT is a summarization model with BERT as both encoder and decoder. We make BERT (an encoder model) work in a seq2seq setting. Then, we need to fine-tune the model to train these connections and the language model head weights. We use the Huggingface's Seq2Seq Trainer object to fine-tune the model using the Seq2SeqTrainingArguments() arguments. The results are calculated using the ROUGE score metric. In our project, we fine-tuned the BERT2BERT base model using a subset of the training dataset due to computational restrictions. For this reason, we also considered the pre-trained BERT2BERT model on huggingface ('patrickvonplaten/bert2bert_cnn_daily_mail') which was already finetuned on the entire CNN dataset.

VI. Graphical User Interface Application

We created a graphical user interface (GUI) application to demonstrate our summarization models. This application was created using Anvil and Google Colab. Anvil is a platform for building and hosting full-stack web apps written entirely in Python. The server was run on Google Colab, as we required GPU computation for the summarization task. The GUI application is as shown in 1.

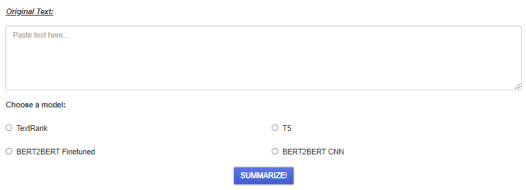

Fig. 1: Summarization GUI

VII. Libraries Used

- **Numpy** : NumPy is the fundamental package for scientific computing with Python. It provides a high-performance multidimensional array object and tools for working with these arrays.
- **Pandas** : Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. It offers a range of data structures and procedures for working with time series and numerical data.
- **Matplotlib** : Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.
- **Transformers** : The transformers library was used which is maintained by Huggingface. Transformers (formerly known as pytorch-transformers and pytorch-pretrained-bert) provides general-purpose architectures (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet…) for Natural Language Understanding (NLU) and Natural Language

Generation (NLG) with over 32+ pretrained models in 100+ languages.

- **Pytorch** : Torch: PyTorch is a Python package that provides two high-level features. Tensor computation (like NumPy) with strong GPU acceleration and deep neural networks built on a tape-based autograd system. TensorData and DataLoader functions were used to convert the data into vectors and tensors.
- **NLTK** : NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries. It was used during the preprocessing step before training the text summarizing models.
- **SimpleT5** : SimpleT5 is a library that is built on top of the PyTorch-lightning and Transformers libraries. It lets you quickly train T5 models with simpler syntax.
- **Gensim** : Gensim is an open-source library that uses contemporary statistical machine learning to do unsupervised topic modeling, document indexing, retrieval by similarity, and other natural language processing functions. For better performance, Gensim is written in Python and Cython.

VIII. Evaluation Metrics

Since the project aims text to text input-output the evaluation metric used was Rouge scores. Recall-Oriented Understudy for Gisting Evaluation also known as Rouge is a metric used for the evaluation of automatic summarization and machine translation. The following types of Rouge metrics were used for evaluation:

For the given reference summary and a generated summary

A. *Rouge Precision*

Rogue precision is the ratio of the overlapping words between both the summaries and the total words in the generated summary.

$$\frac{number\_of\_overlapping\_words}{number\_of\_words\_in\_system\_summary}$$

B. *Rouge Recall*

Rouge recall is the ration of the overlapping words between both the summaries and the total words in the reference summary.

$$\frac{number\_of\_overlapping\_words}{number\_of\_words\_in\_reference\_summary}$$

C. *Rouge F-1*

Rouge F-1 is similar to regular F-1 scores as it is computed using the same formula involving precision and recall. Rouge F-1 is given by

$$\frac{2 * (Precision * Recall)}{Precision + Recall}$$

The Precision and Recall here refer to the Rouge Precision and Rouge Recall.

D. *Rouge-1*

Rouge-1 score refers to the scores when Precision, Recall and F-1 scores are computed using matching unigrams between the summaries.

E. *Rouge-2*

Rouge-1 score refers to the scores when Precision, Recall and F-1 scores are computed using matching bigrams between the summaries.

F. *Rouge-L*

Rouge-1 score refers to the scores when Precision, Recall and F-1 scores are computed using matching the longest common subsequence in the summaries.

IX. Analysis and Results

The comparison of extractive and abstractive summarization, as well as a model that has been fine-tuned on the complete dataset against a subset of the dataset, are the two main focuses of this examination of the results.

The comparison of the Rouge-1, Rouge-2 and Rouge-L scores are displayed in tables I, II, III respectively. The tables compare the Rouge scores for all the 4 models for text summarization.

Since the summary generated by extractive summarization contains the sentences from within the summary, the generated summary was not able to gather all the important info contained within the article to be summarized. The Rouge-1 and Rouge-2 scores are lowest for the summarization done using this method. The abstractive summarization was done using T5, and BERT2BERT models and the abstractive methods performed better in most of the summarizing aspects as denoted by the Rouge Scores. The only exception in which extractive summarization performed better than abstractive summarization is when we compare the Rouge-L scores for the Fine-tuned BERT2BERT model and TextRank. The reason behind this could be that the lesser training data used for fine-tuning BERT2BERT. As evident from the table, when BERT2BERT is fine-tuned on the entire dataset instead of a subset of the dataset, it performs better than TextRank algorithm. Overall the BERT2BERT model finetuned on the entire dataset performs well however T5 which has been finetuned on a subset of the dataset excels in some cases.

One unique observation from the tables is that the Rouge-2 scores are comparatively lesser than Rouge-1 and Rouge-L for all the models. This might be due to the absence of specific bigrams in the generated summaries as compared to reference summaries.

The comparison charts for Rouge-1, Rouge-2, Rouge-L are shown in the figures 2, 3, 4 respectively.. The comparison charts support our analysis of abstractive being better than extractive for summarization purposes in most cases.

TABLE I: ROUGE-1 Scores

	ROUGE-1		
	Precision	Recall	F-measure
TextRank	0.22	0.23	0.22
T5	0.32	0.38	0.34
BERT2BERT Finetuned	0.22	0.23	0.22
BERT2BERT CNN	0.38	0.43	0.40

TABLE II: ROUGE-2 Scores

	ROUGE-2		
	Precision	Recall	F-measure
TextRank	0.03	0.05	0.05
T5	0.12	0.15	0.13
BERT2BERT Finetuned	0.03	0.03	0.03
BERT2BERT CNN	0.17	0.19	0.17

TABLE III: ROUGE-L Scores

	ROUGE-L		
	Precision	Recall	F-measure
TextRank	0.20	0.21	0.20
T5	0.30	0.36	0.32
BERT2BERT Finetuned	0.14	0.15	0.14
BERT2BERT CNN	0.25	0.29	0.26

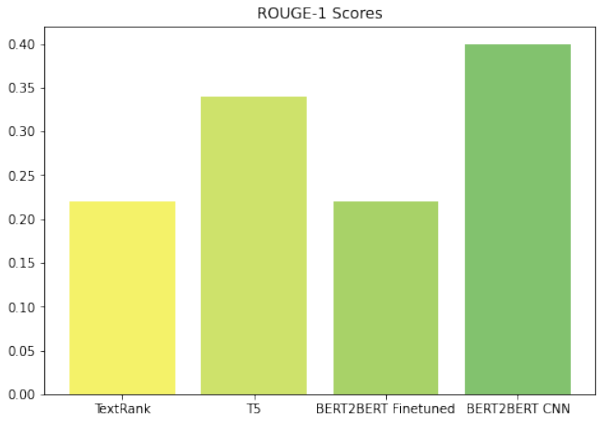

Fig. 2: ROUGE-1 Scores

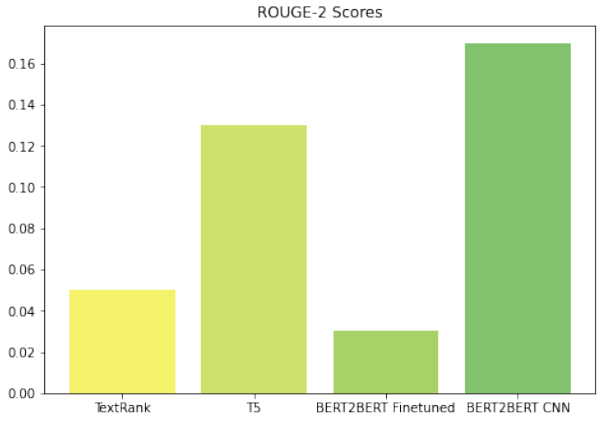

Fig. 3: ROUGE-2 Scores

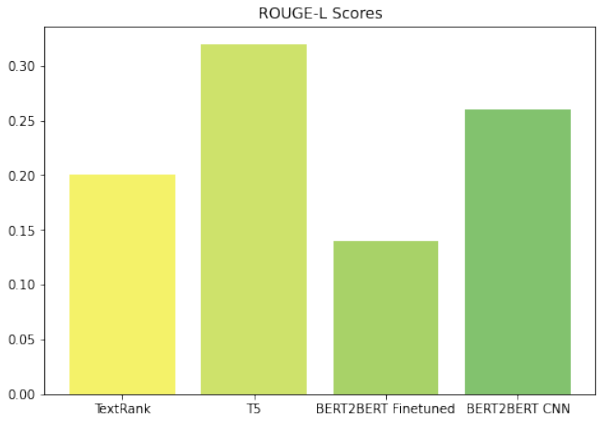

Fig. 4: ROUGE-L Scores

X. Computational Resources

The computational resources required to work on this project were:

- **CPU** : 12th gen Intel(R) Core(TM) i5-1235U 1.30 GHz
- **RAM** : 8.00 GB
- **GPU** : Google Colab GPUs, Kaggle GPU P100

XI. Conclusion

From this project, we can conclude that while extractive methods for summarization can provide good results, abstractive methods typically generate better human-like summaries. We were able to generate meaningful summaries with the T5 and BERT2BERT models, with our best model being the fine-tuned BERT model on the entire CNN dataset. Since all the models are Seq2Seq consisting of encoder and decoder units, the training required high amount of computational resources. We might have gotten better results by increasing the training data size and number of epochs, but this would have required more computational power like GPUs.

XII. Acknowledgment

We'd like to acknowledge Professor Faranak Abri humbly. This project would not have been possible without her constant guidance and support.

References

[1] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang *et al.*, "Abstractive text summarization using sequence-to-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, 2016.

[2] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer." *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.

[3] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 328–11 339.

[4] M. Ravaut, S. Joty, and N. F. Chen, "Summareranker: A multi-task mixture-of-experts re-ranking framework for abstractive summarization," *arXiv preprint arXiv:2203.06569*, 2022.