COMPARATIVE ANALYSIS OF TEXT SUMMARIZATION TECHNIQUES: A FOCUSED CASE STUDY ON THE CNN/DAILYMAIL DATASET

ECE684 (NATURAL LANGUAGE PROCESSING) FINAL PROJECT REPORT

Lavsen Dahal (NetID: ld258)
Department of ECE
Duke University
Durham, NC 27708
lavsen.dahal@duke.edu

Zion Sheng (NetID: zs144)
Department of ECE
Duke University
Durham, NC 27708
zion.sheng@duke.edu

ABSTRACT

This study compares and contrasts different text summarization methods using the CNN/Daily Mail dataset as the main example. We investigate four different approaches for two summarization paradigms: the conventional TF-IDF method and a DL-based BERTsum method for extractive-style summarization; a pre-tuned BART model and a fine-tuned variant of T5-small for abstractive-style summarization. Based on ROUGE metrics, our analysis shows that although the TF-IDF approach offers a basic level of summarization, it is not very effective at understanding context. Better performance is shown by the deep learning models, especially the BART model that was pre-tuned on the same dataset. These models manage to strike a balance between demanding computational resources and sophisticated summarization capabilities. The study demonstrates the improvements in natural language processing and emphasizes the effectiveness of transformer-based models such as BART in managing complicated summarization tasks. Code available at https://github.com/zs144/23fa-ece684-final-proj

Keywords Natural Language Processing · Text Summarization · Deep Learning · Text Encoding · Sequence Generation

1 Introduction

In the field of information processing, text summarization is a crucial method for reducing large amounts of textual content to a more manageable and comprehensible format. By condensing the important information and eliminating the unnecessary details, this procedure guarantees that the original material is preserved. The introduction of automatic text summarization represents a major advancement as it provides an alternative to the tedious and time-consuming process of manual condensation.

Automatic text summarization can be extractive, abstractive or hybrid. In the extractive method, key sentences from the original documents are located and extracted and chosen phrases are put together to create a coherent summary. Conversely, the abstractive method uses a different approach in which the source document or documents are first transformed into a intermediate representation. Next, using this representation, a summary is produced with sentences that are different from those in the source text. A hybrid strategy combines the ideas of abstractive and extractive techniques El-Kassas et al. [2021].

In this work, we used three methods for the purpose of text summarization. Method 1 establishes a foundational approach through its use of the TF-IDF framework which is straightforward and efficient for extractive summarization, but it falls short in terms of comprehending intricate contexts. Summaries that are contextually rich and nuanced are produced by Methods 2 and 3, which use sophisticated deep learning models such as BERTsum(Liu [2019]), T5-smallRaffel et al. [2020], and BART Lewis et al. [2019]. They do, however, come with a trade-off between advanced text summarization capabilities and greater computational resources and implementation complexity due to their sophistication.

2 Related Work

2.1 Traditional Approaches

Generally, there are three types of approaches in the traditional practice of text summarization: statistical-based methods, graph-based methods, and machine-learning-based methods (excluding deep learning). Most traditional methods are for extractive summarization tasks as tackling the abstractive paradigm is not feasible until the emergence of the modern deep learning revolution. Statistical-based methods rely on meaningful statistical features of the input text to score the sentences and extract part of them as the summary. Radev et al. [2004] uses cosine similarity to the centroid sentence as the key feature, where the centroid sentence is defined as the one with the highest average TF-IDF score. Afsharizadeh et al. [2018] propose a more complicated way to compute the sentence score with up to 11 features, including sentence positions, topic frequency, start cluster frequency, etc. Then, sentences are selected from the one that scores the highest and then keep moving down until the summary reaches the length limit.

Graph-based methods use graph models to abstract a given article into nodes and edges. Sentences are represented as nodes, while sentence similarity values are assigned to the edges as weights. Relative studies in this area usually focus on designing new methods to measure sentence importance/similarity or some new mechanisms to represent the relationship between sentences. Mihalcea and Tarau [2004] introduces TextRank algorithm, which uses the concepts of "voting"/"recommendation" to build a graph-based ranking model. It is proven to work well on keyword extraction and text summarization tasks. Another classic model is LexRank, proposed by Erkan and Radev [2004]. LexRank chooses to use eigenvector centrality to measure the sentence importance in a graph representation of sentences, and is verified to perform well in extractive summarization tasks as well.

Some traditional ML-based methods are also worth mentioning. Fuentes et al. [2006] presents the application of Support Vector Machine (SVM) for query-based summarization using information from pyramids of summary content units. The model introduced by Conroy and O'leary [2001] uses Hidden Markov Chain (HMM) to determine the likelihood of whether a sentence should be included in the summary. Kupiec et al. [1995] proposed an extractive summarizer based on Bayes rules. The model computes the probability of a sentence being included in the summary under the condition that features equal to a certain value. These features include sentence length cut-off feature, paragraph feature, etc. All methods mentioned here claim to achieve a competitive performance on DUC systems.

2.2 DL-based Approaches

Although the traditional approaches paved the way in the early stage of text summarization research, recent studies have been dominated by deep learning-based methods. These methods are preferred as they can learn patterns from data automatically, which tends to outperform traditional methods that rely on features pre-set by humans. In fact, many well-known DL architectures work amazingly well on text summarization tasks after some simple modifications. For instance, Nallapati et al. [2017] apply two bidirectional RNN for text encoding and sentence extracting respectively based on the distinct scores for salience, location, and novelty of the sentences. Cheng and Lapata [2016] employ CNN layers to process the document input at the word level along with the RNN layers to encode the text at the sentence level. Then, the outputs from the encoder are passed to a sentence extractor leveraging the neural attention mechanism. By combining more components, such as LSTM/GRU, RNN, and attention mechanism, more opportunities are unlocked. Zhou et al. [2018] use several NLP modeling techniques and design a compound model architecture called NEUSUM, which can jointly score and select sentences for the summary.

Transformer-based methods, which use self-supervised learning techniques, have greatly influenced recent advances in abstractive summarization Lewis et al. [2019], Raffel et al. [2020], Zhang et al. [2020]. Transformers use attention mechanisms to determine how important each component of the input data is. Transformers are capable of discovering intricate patterns and relationships in the data when they are used for self-supervised tasks like guessing missing words or sentences. Generally, there are two stages to this learning process: pre-training and fine-tuning. The model uses a large unlabeled dataset to learn general language patterns during pre-training. The model is then fine-tuned using a smaller, task-specific dataset to carry out particular tasks like summarization, translation, or question answering.

3 Methods

3.1 A traditional extractive summarizer (TF-IDF)

This method serves as the baseline for this study as it derives a very simple but working extractive summarizer. It utilizes a key concept called TF-IDF (Term Frequency - Inverse Document Frequency), which we have learned in ECE 684. TF-IDF scores a word by multiplying TF (term frequency in the corpus) and IDF (inverse document frequency).

Common practices for smoothing, such as taking the logarithm and adding one to the denominator, are also applied here. Suppose there are M articles $(\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_M)$, the TF-IDF score for word w in article $\mathbf{d}$ can be expressed as:

$$\text{TF-IDF}(w) = \text{DF}(w) \times \text{IDF}(w), \tag{1}$$

$$\text{DF}(w) = \frac{\text{count}(w, \mathbf{d})}{\sum_{w_i \in \mathbf{d}} \text{count}(w_i, \mathbf{d})}, \quad \text{IDF}(w) = -\log \frac{1}{M}(1 + \sum_{i=1}^{M} \mathbb{I}(w \in \mathbf{d}_i)), \tag{2}$$

where $\text{count}(w, \mathbf{d})$ is the raw count of how many times the word w occurs in document $\mathbf{d}$. $\mathbb{I}(w \in \mathbf{d}_i)$ is an indication function which equals 1 when the word w appears in document $\mathbf{d}_i$, or it equals 0 otherwise. A higher TF-IDF score indicates that the word is more important to the document as it appears frequently in the given text exclusively. Based on this, we can compute a score for each sentence by averaging the TF-IDF scores of all the words it contains. To extract the best subset of sentences to represent the whole article, we can select those with the highest sentence scores. The sentence score is higher when it contains more important words, so sentences with the highest score should carry the most important information for the whole text. This method is very convenient to implement as it does not require any training process before deployment for inferences. However, the drawback is that it needs at least two documents in the input corpus to derive a meaningful IDF.

3.2 A DL-based extractive summarizer (BERTsum)

For method 2, the plan is to explore some more sophisticated but powerful deep learning architecture for extractive summarization. Since transformer-based methods have proven to be highly effective in text summarization tasks, we decided to choose one from this genre as the representative. Among all the existing choices, BERTsum (Liu [2019]) seems to be the best starting point, as it is both easy to understand and implement. The basic model architecture can be visualized using the following diagram:

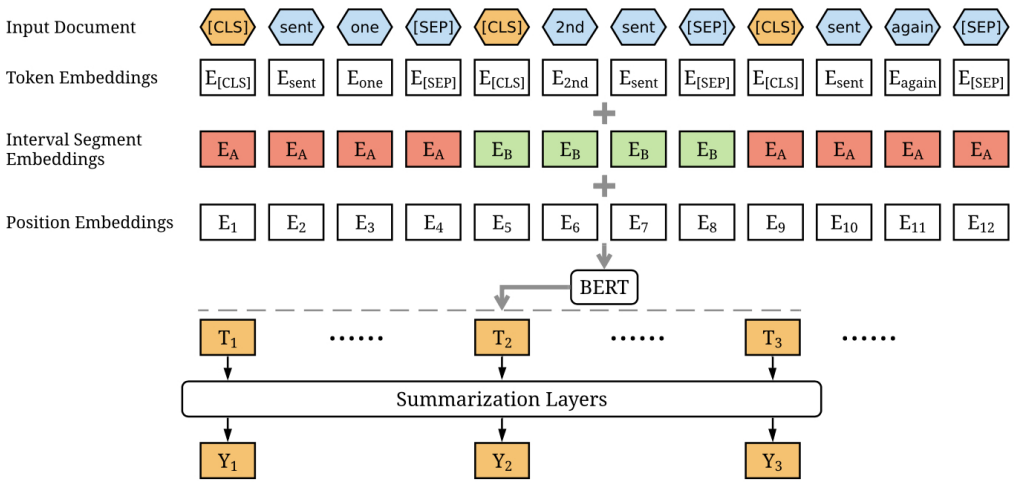


Figure 1: The overview architecture of the BERTsum (credit: Liu [2019])

The extractive summarization task can be formalized as: Given a document consisting of several sentences $\mathbf{d} = [s_1, s_2, ..., s_M]$, each sentence s_i will be assigned with a label $y_i \in \{0, 1\}$ indicating whether the sentence should be included in the summary or not. BERTsum, as its name suggests, utilizes the pre-trained BERT model for the sentence selection process by generating the representation for each sentence. However, we need to tailor the original BERT to match the requirements here. Firstly, the output of the original BERT is grounded to tokens rather than sentences; Secondly, the segmentation embedding layer in the original version only has two labels, but we need more labels to differentiate multiple sentences in the article. To solve these problems, Liu [2019] proposed some effective modifications as shown in Figure 1. To encode multiple sentences, we insert a pair of `[CLS]` and `[SEP]` tokens before and after each sentence. These tokens can aggregate features for sentences. Furthermore, to distinguish multiple sentences in the article, we use two embedding layers, the interval segment embedding layer and the positional embedding layer, to encode the positional information. For each sentence s_i, the output vector of the corresponding `[CLS]` token from the `BERT` layer will be used as the representation.

There are several available choices for the `Summarization` layer. To achieve a better performance, we choose to use a unidirectional LSTM layer as recent studies have confirmed the synergy between transformers and RNN-based techniques (Chen et al. [2018], Wang et al. [2019]). The output T_i from the BERT layer is fed as the input to the LSTM cell at time step i. Then, the corresponding output of the LSTM cell can be expressed as:

$$F_i, I_i, O_i, G_i = \text{LN}_h(W_h h_{i-1}) + \text{LN}_x(W_x T_i), \tag{3}$$

$$C_i = \sigma(F_i) \odot C_{i-1} + \sigma(I_i) \odot \tanh(G_{i-1}), \tag{4}$$

$$h_i = \sigma(O_i) \odot \tanh(\text{LN}_c(W_c C_i)), \tag{5}$$

where F_i, I_i, O_i, G_i are the values returned from the forget gate, input gate, and output gate respectively; G_i, C_i are hidden vectors and memory vectors accordingly; $\text{LN}_h, \text{LN}_x, \text{LN}_c$ are layer normalization layers (Ba et al. [2016]). h_i is the output vector, which will be passed through a final linear layer and a sigmoid activation function to get the final prediction result:

$$\hat{Y}_i = \sigma(W_o h_i + b_o), \tag{6}$$

where W_o, b_o are the weight and bias term in the final linear layer.

3.3 DL-based abstractive summarizers (BART and T5-small)

In this section, we describe transformer-based models for text summarization, specifically focusing on the T5-small and BART models. The T5 (Text-To-Text Transfer Transformer) model uses an encoder-decoder architecture and a special "blank-filling" pre-training task to understand and predict text, standardizing all NLP tasks into a text-to-text format. However, BART (Bidirectional and Auto-Regressive Transformers) is better at text reconstruction tasks because it combines autoregressive text generation (like GPT) with bidirectional context comprehension (like BERT). BART focuses specifically on comprehending and producing text, in contrast to T5's uniform approach to task formulation. Although T5 is flexible and reliable for a wide range of tasks, BART excels at more difficult text generation and transformation jobs.

The BART model, pre-trained and fine-tuned on the CNN/Daily Mail dataset, was utilized directly for inference using resources from the Hugging Face repository. This allowed us to evaluate its summarization capabilities without additional training. Conversely, the T5-small model was initially used in its pre-trained state, undergoing a two-stage process. Firstly, we conducted inference without any task-specific fine-tuning to establish a baseline performance. Subsequently, we implemented a fine-tuning strategy, training the T5-small model specifically on the CNN/Daily Mail dataset. This approach allowed us to directly compare the impact of fine-tuning on the model's summarization effectiveness.

4 Experiment

4.1 Data

CNN/DailyMail Dataset (Hermann et al. [2015]) was first created for the research in machine reading and question answering, but later adapted for the text summarization task by Nallapati et al. [2016]. It contains over 300k unique news articles published by CNN (93k) and the Daily Mail (220k). The articles are all written in English, and each article is paired with a human-generated summary. The dataset can be directly used on abstractive summarization tasks as the given summaries are written in their own words. We download the dataset provided by See et al. [2017], which can be accessed from many places, such as HuggingFace and Kaggle. For extractive summarization, we first need to generate some extractive summaries as the replacement for the original summaries. Housen [2023] implemented a generic script that can create a completely extractive summary by maximizing the ROUGE score between itself and the ground-truth abstractive summary. The script and pre-processed data can be downloaded from the supplementary GitHub repository maintained by Housen [2023], where the data has already been splitted into three sets: training set (287,113 articles), validation set (13,368 articles), and test set (11,490 articles), followed by the standard processing from Hermann et al. [2015].

4.2 Implementations

Method 1 is implemented by `scikit-learn` and `nltk`. We use the function `sent_tokenize()` from `nltk` to tokenize the sentences in each article, and then pass them to a `TfidfVectorizer` object initialized with `stop_words`

downloaded from `nltk`. After averaging, we will get all the sentence scores. Method 1 is very easy to implement, but the downside of the simplicity is that the algorithm does not have any intelligence to understand the context. Therefore, we need to do more pre-processing operations, including: (1) removing the repetitive meaningless sentences and overly short sentences; (2) setting the sentence count in a summary to a fixed number. We use the average length, which is 4.

Method 2 is implemented by PyTorch. Instead of the original BERT model, we choose to use a better version called DistilBERT. This better version shares the same architecture as the original one, but uses a technique called knowledge distillation to effectively prune the redundant parts, ending up with a 40% smaller size but remaining 97% of its language comprehending capabilities and also being 60% faster (Sanh et al. [2019]). We download the pre-trained DistilBERT model from HuggingFace, and plug it into our BERTsum algorithm. Thanks to the standardized modeling pipeline provided by Housen [2023], the training/testing process can be easily implemented and conducted. We use Adam optimizer to fine-tune the model with an initial learning rate of 0.001, and momentum parameters $\beta_1 = 0.9, \beta_2 = 0.999$. The model is trained for $50,000$ epochs on GPU provided by Google Colab.

Method 3 used two different transformer architectures in our study: T5_small and BART. We obtained these models by downloading their pre-trained versions from the HuggingFace repository. We used the same data splits as suggested by See et al. [2017] for training, validation, and testing in order to be consistent with earlier research. We specifically altered the input data for the T5_small model's fine-tuning process by adding the prompt "summarize:" to each article in the CNN/Daily Mail dataset. This method is consistent with the T5 model's intended application to text summarization tasks. The 'highlights' section of the dataset was used as a summary target for the fine-tuning. To ensure that there was enough context for summarization without overloading the model, we used the tokenizer of the T5_small model to convert text to tokens, with a maximum sequence length of 512 tokens. Shorter sequences were padded with a PAD token in order to maintain consistent sequence lengths for effective batch processing in neural networks. In order to prevent memory problems and preserve optimal model performance, longer sequences that exceeded the 512-token limit were truncated.
The BART model is well-suited for summarization tasks involving comparable kinds of data because it was pre-trained and refined on the CNN/Daily Mail dataset. In this study, we directly carried out inference on the dataset using this refined model. By using the model's current capabilities to effectively produce summaries, we avoided additional training.

4.3 Evaluation Metrics

In this study, we choose to use ROUGE (Recall-Oriented Understudy for Gisting Evaluation) as the main metric to measure the quality of the summarization. There are multiple variations in the ROUGE family. We choose to use the most commonly used ones: ROUGE-1, ROUGE-2, and ROUGE-L (Lin and Hovy [2003]). Each of these variations computes the similarity between the reference (ground truth summary) and the candidate (generated summary). The N in ROUGE-N refers to the direct N-gram overlaps between the candidate and the reference with the caveat that ROUGE-L computes the longest common sub-sequence between the two. For any ROUGE-N, the ROUGE score can be computed in three different ways depending on the denominator in the expression. Take ROUGE-2 for instance, these are:

$$\text{ROUGE-2} = \frac{\text{shared bi-grams}}{\text{\# of bi-grams in reference}} \qquad \text{(recall)}$$

$$\text{ROUGE-2} = \frac{\text{shared bi-grams}}{\text{\# of bi-grams in candidate}} \qquad \text{(precision)}$$

$$\text{ROUGE-2} = 2 \cdot \frac{\text{recall} * \text{precision}}{\text{recall} + \text{precision}} \qquad \text{(F-1)}$$

We choose to use the F-1 version for all ROUGE scores as it combines both recall and precision, leading to a more balanced measurement. In the later sections of this report, all ROUGE scores refer to their F-1 score without further specification. From the above expressions, the ROUGE score is between 0 and 1. A higher score indicates a higher similarity between the candidate and the reference. Specifically, ROUGE-1 and ROUGE-2 are reported to measure the formativeness of the generated summaries, whereas ROUGE-L is employed alongside ROUGE-1 and ROUGE-2 to provide a more nuanced evaluation of summarization systems, taking into account factors such as content order, longer phrases, and abstraction.

It's important to note that we conduct both extractive and abstractive summarization tasks on the same dataset, CNN/DailyMail Dataset. In other words, the reference (ground truth summary) remains the same for all methods. Even for the extractive summarizers derived in method 1 and method 2, their ROUGE score is computed with the abstractive summaries provided originally. The goal of the study is to identify the best text summarizer considering not

only its inference accuracy (ROUGE scores) but also other factors like training cost, inference speed, model size, and extensibility on different datasets. Therefore, it is imperative to ensure that the comparison is fair and unbiased.

5 Results And Discussion

In this section, we present the experimental results and provide some analysis. The content is structured into two parts. The first part, inner-category comparison, focuses on comparing methods within the same summarization paradigm. Specifically, we compare method 1 (TF-IDF) and method 2 (BERTsum), which all belong to extractive summarization, as well as method 3-1 (BART) and 3-2 (T5-small), which are solutions we tried for abstractive summarization. Next, in the second part, we compare the best candidate from the two sides in various aspects to derive a comprehensive observation that which one is more competitive for the text summarization task. Due to the limited space here, we will put some tables and graphs in the Appendix section at the end of the report.

5.1 Inner-Category Comparison

For the two methods we explored for extractive summarization, method 2 (BERTsum) performs significantly better than method 1 (TF-IDF) as shown in the subplot (a) and (b) in Figure A.1. The figure summarizes the distributions of ROUGE scores from different methods tested on the testing set of CNN/DailyMail and displays them in several histograms. The two distributions, method 1 and method 2, both have a bell-shaped curve. However, the peak of method 2 always appears at a higher point than that of method 1. In other words, the overall ROUGE scores from method 2 are higher than method 1, and this is also reflected by the gap between their average levels. This is not surprising as DL-based summarizers tend to outperform traditional methods in extractive summarization as we have seen in the previous studies. Despite the huge differences between the algorithms, the success of DL-based summarizers (including method 2 here) depends on their ability to comprehend the entire article, rather than taking each sentence into account without the context. This gives DL-based summarizers a vast advantage in finding the set of sentences that can maximize the carried information, and we believe BERT is even one of the best competitors among those DL-based methods as it processes the text from two directions, which further enhances the understanding.

However, the qualitative result gives us more insights into their performances. For some instances, such as the first article in Appendix A.1, the summary generated by method 1 and method 2 both contains repetitive information. This is because the summarizers are asked to maximize the captured information with a limited sentence/word count budget. This optimization goal will distort its outcomes by encouraging the summarizers to keep selecting the sentences that contain the core information, without considering that they may just repeat similar things. Although the information is important to the article, the repetitiveness does affect the reading experience for human readers. Still, the summary from method 2 is more reasonable since it maintains the original order. In contrast, the summary from method 1 changes the order and creates confusion regarding the referent of some pronouns.

Similar results were obtained when we compared two T5_small tokenizer variants on the CNN/Daily Mail dataset: the T5 tokenizer and the auto tokenizer from HuggingFace. On the other hand, T5_small's ROUGE-1 score increased from 0.33 to 0.38 after fine-tuning with CNN/DailyMail dataset as shown in Figure A.1 and A.2. The architecture and task-specific optimization of BART, which was pre-trained and refined on this dataset, allowed it to outperform T5_small fine-tuned. The superior performance of BART is ascribed to the alignment of its training data and its architecture, which efficiently blends autoregressive and bidirectional transformers for summarization tasks.

5.2 Cross-Category Comparison: Extractive VS Abstractive

Since the ultimate goal is to generate a fluent and succinct summary for any input articles, it is necessary to find the best out of the best, i.e., determine the best one between the best candidates from the two categories. From the previous analysis, the best candidate for the extractive summarization category is BERTsum, while the counterpart for abstractive summarization is BART. Figure 2 shows that ROUGE scores of BART are slightly higher than BERTsum. We argue that the modest advantage mostly comes from factors other than differences in model architecture since both models are Transformers and do not differ significantly. One possible reason could be the systematic benefits of abstractive summarization for some of the input texts that are just unfriendly for extractive summarizers, such as the first article we presented in Appendix A.1. By comparing the self-created summaries generated from method 2 (BERTsum) and method 3 (BART), we can notice that the latter returns a more coherent and informative diverse summary, whereas method 2 selected several sentences with duplicated information. Nevertheless, such a benefit does not exist everywhere. For the second article in Appendix A.1, the summary generated by BERTsum scores higher and conveys the key information without much redundancy. On the contrary, the summary from BART is not perfect this time as it contains some incoherent sentences in the end.

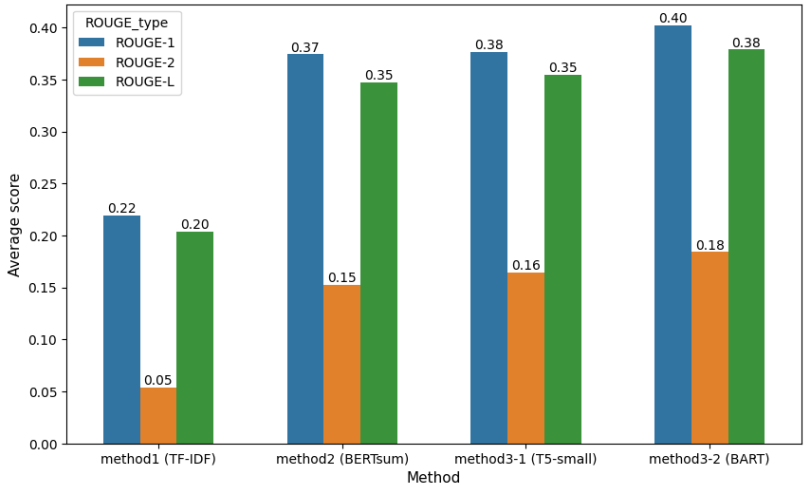


Figure 2: Comparing average ROUGE scores from different methods on CNN/DailyMail testing set

Other factors aside from the ROUGE scores should be considered as well. Although BERTsum, T5-small, BART are all Transformers, the differences in model architectures can directly affect some important performance aspects, such as the model size and training/inference time efficiency. For instance, it takes about 1 minute for BERTsum to compose the summary for one instance, but the time needed by T5-small is only a few seconds. Another important thing to investigate is whether the model can be generalized to the text data of different topics or languages. Even if the models work well on CNN/DailyMail dataset, it is uncertain that the good performance can be reproduced in some other domains, like medical reports or academic literature written in Spanish. Furthermore, we notice the relative competitiveness of these summarizers is not unchanged. Despite the pattern unveiled by the general statistical analysis we have done so far, it is also thought-provoking to study some special instances where certain methods accidentally fail. This is true even for the state-of-the-art methods. We believe that these topics are worth exploring, but we will leave them for future investigations.

6 Conclusion

In this study, we assessed several different approaches, TF-IDF, BERTsum, BART, and T5-small, in our review of extractive and abstractive summarization strategies on CNN/DailyMail dataset. While each approach has advantages, our results show that the BART model, which was customized for this dataset, performed better than the other approaches in terms of ROUGE-1, ROUGE-2, and ROUGE-L scores. This demonstrates how well transformer-based models, such as BART, perform in challenging summarization tasks, thanks to their sophisticated architecture and dataset-specific fine-tuning. It's interesting to note that, despite its simplicity, the TF-IDF method was able to produce summaries; however, these were not as coherent or relevant as those produced by more sophisticated models. This experiment demonstrates how natural language processing has advanced and shows how advanced models can significantly improve the quality of automated text summarization.

References

Wafaa S El-Kassas, Cherif R Salama, Ahmed A Rafea, and Hoda K Mohamed. Automatic text summarization: A comprehensive survey. *Expert systems with applications*, 165:113679, 2021.

Yang Liu. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

Dragomir R Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. Centroid-based summarization of multiple documents. *Information Processing & Management*, 40(6):919–938, 2004.

Mahsa Afsharizadeh, Hossein Ebrahimpour-Komleh, and Ayoub Bagheri. Query-oriented text summarization using sentence extraction technique. In *2018 4th international conference on web research (ICWR)*, pages 128–132. IEEE, 2018.

Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004.

Günes Erkan and Dragomir R Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479, 2004.

Maria Fuentes, Enrique Alfonseca, and Horacio Rodríguez. Support vector machines for query-focused summarization trained and evaluated on pyramid data. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions - ACL '07*, Dec 2006. doi:10.3115/1557769.1557788. URL `http://dx.doi.org/10.3115/1557769.1557788`.

John M Conroy and Dianne P O'leary. Text summarization via hidden markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 406–407, 2001.

Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 68–73, 1995.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*, 2016.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. Neural document summarization by jointly learning to score and select sentences. *arXiv preprint arXiv:1807.02305*, 2018.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*, 2018.

Zhiwei Wang, Yao Ma, Zitao Liu, and Jiliang Tang. R-transformer: Recurrent neural network enhanced transformer. *arXiv preprint arXiv:1907.05572*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.

Hayden Housen. Transformersum. `https://github.com/HHousen/TransformerSum`, 2023.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 human language technology conference of the North American chapter of the association for computational linguistics*, pages 150–157, 2003.

A Appendix

A.1 Some Qualitative Results

Example article #1:
A drunk teenage boy had to be rescued by security after jumping into a lions' enclosure at a zoo in western India. Rahul Kumar, 17, clambered over the enclosure fence at the Kamla Nehru Zoological Park in Ahmedabad, and began running towards the animals, shouting he would 'kill them'. Mr Kumar explained afterwards that he was drunk and 'thought I'd stand a good chance' against the predators. Next level drunk: Intoxicated Rahul Kumar, 17, climbed into the lions' enclosure at a zoo in Ahmedabad and began running towards the animals shouting 'Today I kill a lion!' Mr Kumar had been sitting near the enclosure when he suddenly made a dash for the lions, surprising zoo security. The intoxicated teenager ran towards the lions, shouting: 'Today I kill a lion or a lion kills me!' A zoo spokesman said: 'Guards had earlier spotted him close to the enclosure but had no idea he was planing to enter it. 'Fortunately, there are eight moats to cross before getting to where the lions usually are and he fell into the second one, allowing guards to catch up with him and take him out. 'We then handed him over to the police.' Brave fool: Fortunately, Mr Kumar fell into a moat as he ran towards the lions and could be rescued by zoo security staff before reaching the animals (stock image) Kumar later explained: 'I don't really know why I did it. 'I was drunk and thought I'd stand a good chance.' A police spokesman said: 'He has been cautioned and will be sent for psychiatric evaluation. 'Fortunately for him, the lions were asleep and the zoo guards acted quickly enough to prevent a tragedy similar to that in Delhi.' Last year a 20-year-old man was mauled to death by a tiger in the Indian capital after climbing into its enclosure at the city zoo.

Ground Truth Summary
Drunk teenage boy climbed into lion enclosure at zoo in west India. Rahul Kumar, 17, ran towards animals shouting 'Today I kill a lion!' Fortunately he fell into a moat before reaching lions and was rescued.

Method 1 (TF-IDF)
A police spokesman said: 'He has been cautioned and will be sent for psychiatric evaluation. The intoxicated teenager ran towards the lions, shouting: 'Today I kill a lion or a lion kills me!' Mr Kumar explained afterwards that he was drunk and 'thought I'd stand a good chance' against the predators. A drunk teenage boy had to be rescued by security after jumping into a lions' enclosure at a zoo in western India.

Method 2 (BERTsum)
Rahul Kumar , 17 , clambered over the enclosure fence at the Kamla Nehru Zoological Park in Ahmedabad , and began running towards the animals , shouting he would ' kill them '. Mr Kumar explained afterwards that he was drunk and ' thought I 'd stand a good chance ' against the predators. Next level drunk : Intoxicated Rahul Kumar , 17 , climbed into the lions ' enclosure at a zoo in Ahmedabad and began running towards the animals shouting ' Today I kill a lion ! '.

Method 3 (BART)
Rahul Kumar, 17, clambered over enclosure fence at Kamla Nehru Zoological Park. He ran towards the animals, shouting: 'Today I kill a lion or a lion kills me!' Fortunately, Mr Kumar fell into a moat before reaching the animals. He has been cautioned and sent for psychiatric evaluation.

Method 4 (T5-small)
Rahul Kumar, 17, climbed into enclosure fence at zoo in western india. he ran towards the animals shouting: 'Today I kill a lion or a lion kills me!' zoo guards acted quickly enough to prevent a tragedy similar to that in Delhi.

Table 1: ROUGE scores of the summary generated by different methods for the example article #1

	Method 1 (TF-IDF)	Method 2 (BERTsum)	Method 3 (BART)	Method 4 (T5-small)
ROUGE-1	0.39	0.41	0.49	0.51
ROUGE-2	0.15	0.11	0.27	0.24
ROUGE-L	0.35	0.34	0.47	0.49

Example article #2:
Alan Pardew has warned Crystal Palace's Barclays Premier League admirers that if they want to sign Yannick Bolasie this summer then the bidding will have to start at £20million. It has been suggested that a £10m bid could tempt Palace into selling the winger, who has excelled since Pardew arrived in January and launched the club up the table. But the Palace manager is adamant that the club's leading assets do not want to leave – and any players that do go will not be sold cheaply. That applies particularly to Bolasie, who scored a hat-trick in the demolition of Sunderland last week. Yannick Bolasie (centre) of Crystal Palace celebrates scoring against Sunderland at the Stadium of Light . Alan Pardew says Bolasie bidding will have to start at £20m for his star to leave Selhurst Park . Responding to a question about Joel Ward's contract extension, Pardew said: 'It was good news for us this week (with Ward). We wanted to send a statement that our players will be very, very difficult to take away. 'One, because they enjoy their football and two because we are going forward. Brede Hangeland said to me when he was at Fulham that he sensed that no one wanted to leave. (There was a report of a) £10m price on Bolasie. Blimey. 'It needs to be more than double that to get near him. We are in a great financial position. I am not getting carried away with where we are but we are doing very well in the Premier League.' That form includes four straight wins ahead of Tony Pulis's return to Selhurst Park on Saturday. Pardew believes Pulis should get a good reception at the club which he transformed last season before leaving days before this campaign started. Bolasie scored a hat-trick in Crystal Palace's demolition of Sunderland last week away from home . Pardew is demanding more than £20m for Bolasie to be taken away from Crystal Palace during the summer . But Pardew also indicated that his own job in lifting Palace this season had spared Pulis a potentially rocky return. He said: 'Tony achieved fantastic results here last season. For sure I wouldn't be here and a lot of these players wouldn't be here if he hadn't done that, so we are thankful to him. 'He should also be thankful that we are in the position we are in, and that the players and myself have got the team where it is, because the reception could have been very different if we were in the relegations zone and leaving so shortly before the season started. I think that is good for him and he deserves that. Hopefully tomorrow he gets the reception that he warrants.' Pardew has targeted a top-10 finish, saying: 'Top 10 is a realistic aim and then we want to build on that. The chairman has done a fantastic job. I will be banging his door down in the summer for some big signings.' Bolasie scores against Sunderland as he helped Crystal Palace beat the struggling Premier League side .

Ground Truth Summary
Yannick Bolasie scored a hat-trick against Sunderland last week. The Crystal Palace star is the wishlist of clubs for during the summer. Alan Pardew warns his admirers that Bolasie will cost them at least £20m.

Method 1 (TF-IDF)
I think that is good for him and he deserves that. 'It needs to be more than double that to get near him. For sure I wouldn't be here and a lot of these players wouldn't be here if he hadn't done that, so we are thankful to him. Hopefully tomorrow he gets the reception that he warrants.'

Method 2 (BERTsum)
Alan Pardew says Bolasie bidding will have to start at £ 20 m for his star to leave Selhurst Park. Bolasie scored a hat - trick in Crystal Palace 's demolition of Sunderland last week away from home. Pardew is demanding more than £ 20 m for Bolasie to be taken away from Crystal Palace during the summer.

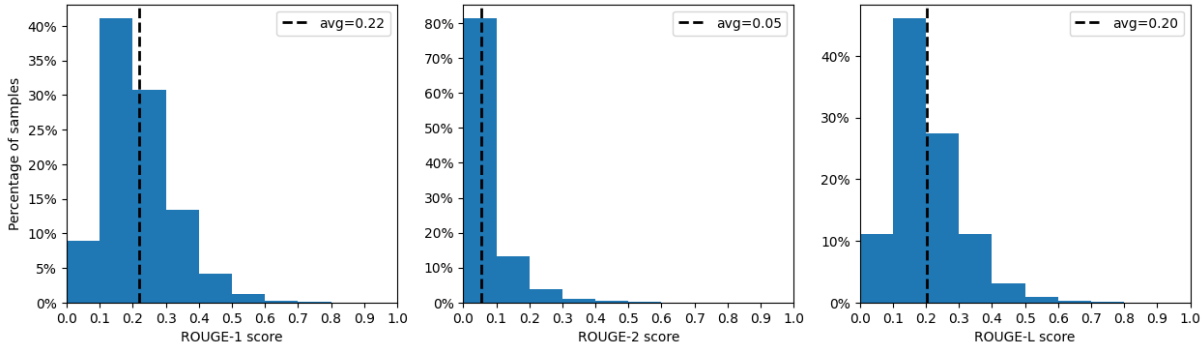
Method 3 (BART)
Yannick Bolasie scored a hat-trick in Crystal Palace's win over Sunderland. Alan Pardew is demanding more than £20m for the winger to leave. The Eagles boss says the bidding will have to start at £20million. Tony Pulis returns to Selhurst Park for the first time since leaving.

Method 4 (T5-small)
a £10m bid could tempt the club into selling the winger. the winger scored a hat-trick in crystal palace's demolition of sunderland last week. however, he is demanding more than £20m for the winger to leave.

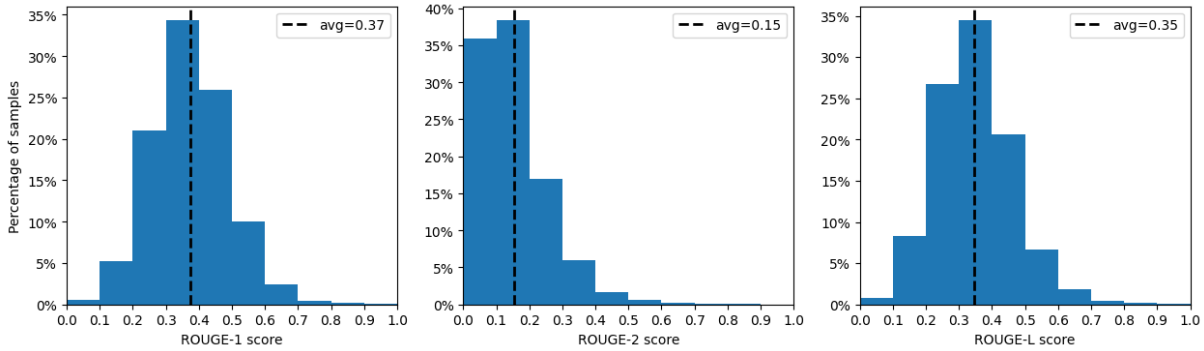
Table 2: ROUGE scores of the summary generated by different methods for the example article #2

	Method 1 (TF-IDF)	Method 2 (BERTsum)	Method 3 (BART)	Method 4 (T5-small)
ROUGE-1	0.16	0.51	0.43	0.32
ROUGE-2	0.00	0.19	0.13	0.09
ROUGE-L	0.16	0.46	0.43	0.32

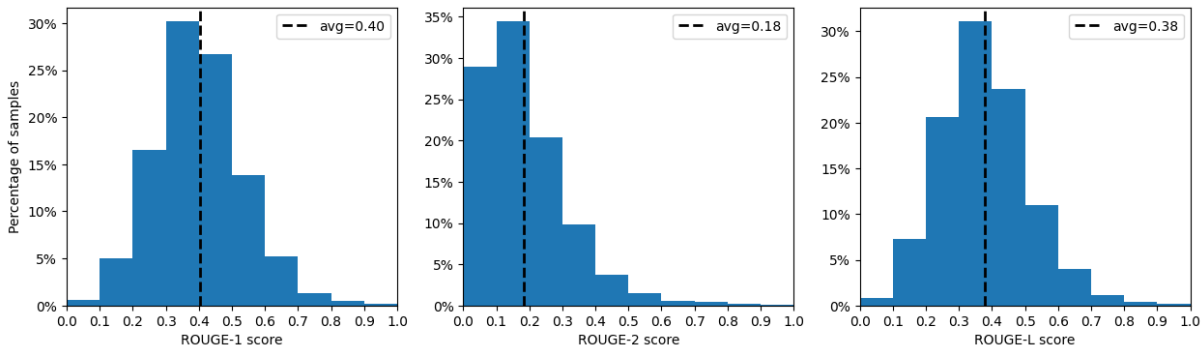
A.2 Some Quantitative Results



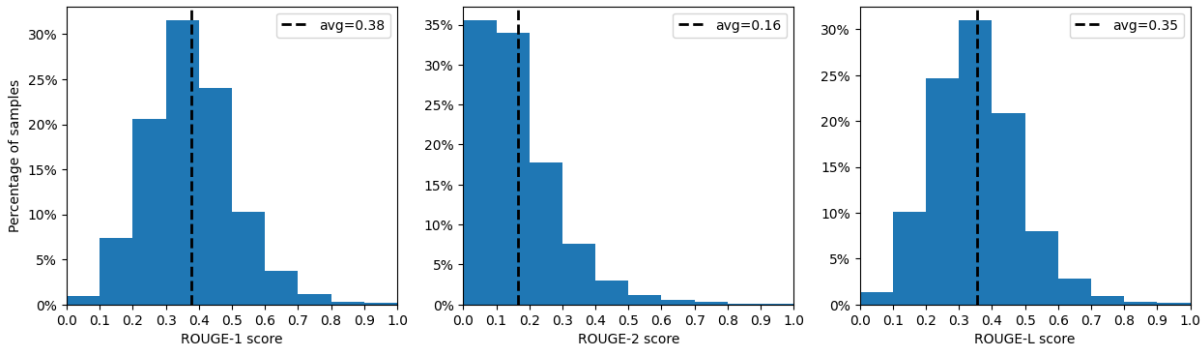
(a) method 1: TF-IDF



(b) method 2: BERTsum

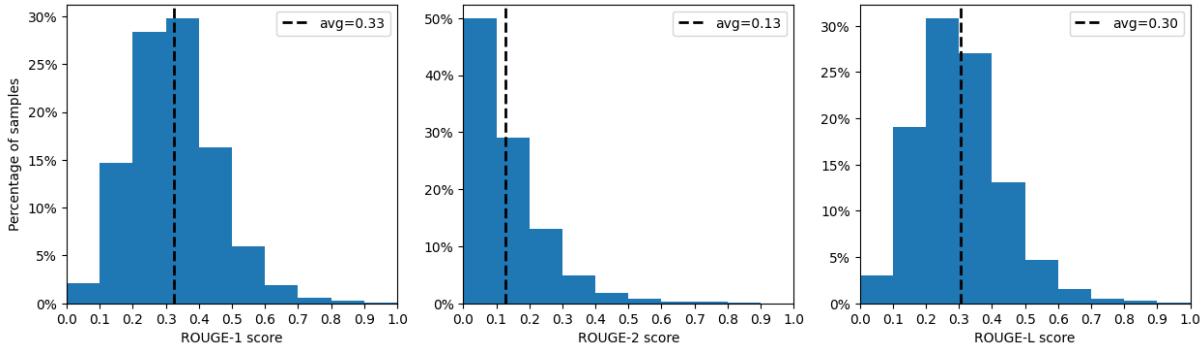


(c) method 3: BART

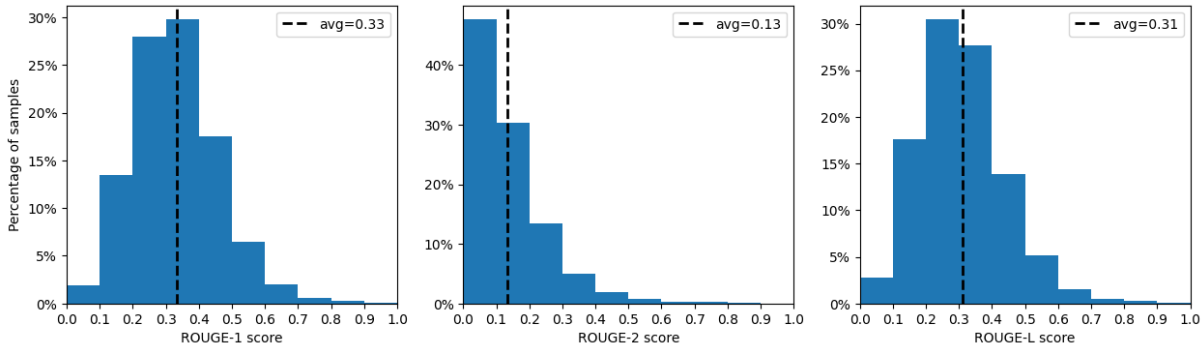


(d) method 4: T5-small

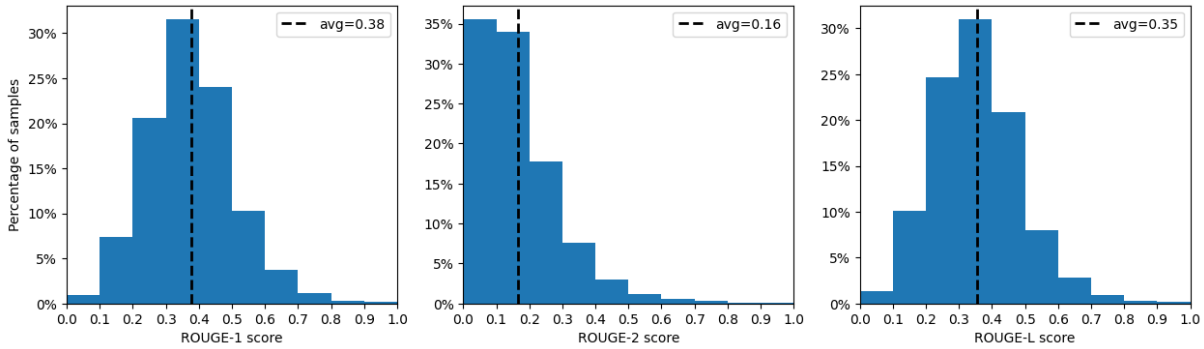
Figure A.1: Distribution of ROUGE scores on CNN/DailyMail dataset from different summarizers



(a) T5-small plain



(b) T5-small with auto tokenizer



(c) T5-small after fine-tuning **[Best]**

Figure A.2: Distribution of ROUGE scores on CNN/DailyMail dataset from different T5-small variants