# ZPJa: Summarization - Methods Comparison

**Bc. Alexander Polok**
`xpolok03@fit.vut.cz`

**Ing. Martin Dočekal**
`idocekal@fit.vut.cz`

## Abstract

In recent years there has been an explosion in the amount of text data from a variety of sources on the internet. This exponentially growing volume is an inestimable source of information and knowledge which needs to be effectively summarized to be useful. This work focuses on the comparison of different extractive and abstractive techniques of generic summarization on a single document. Techniques were evaluated on the CNN/Daily Mail and Wikihow datasets. Currently used methods, architectures, and associated problems are further discussed.

## 1 Task Definition

Text summarization is a technique in natural language processing for constructing a tiny and crisp version of a large textual document while preserving the meaning of the original text document. The increasing volume of documents in the big data era has demanded exhaustive research in the NLP area for automatic text summarization, since creating the summary manually is an extremely tedious job. Summarization may be a multi-document or single document, and generic or query-based. This work only focuses on the generic single-document summarization.

## 2 Methods

Although there are many different approaches to single document summarization they could be split into two main groups based on the process of summarization. Few commonly used approaches will be covered for each group.

### 2.1 Extractive

Extractive summarization (similar to highlighting with a pen) is a method that generates a short summary from the original document by picking the most important sentences from the original document and concatenating them. Before the boom of deep neural networks in recent years, extractive summarization was mainly used for its simplicity and relatively good results. The extractive summary consists of a subset of original sentences, which should be grammatically correct. However, these sentences may suffer from the lack of cohesion[1]. Sentences are extracted and "just" concatenated to each other, it is quite common that there exists no smooth transition between ideas/concepts/topics in different sentences. Another problem that may occur is that some words from the information-rich sentences may contain minimal informative benefit and are still present in the final summary.

### 2.2 First approaches

The positional method (Baxendale, 1958) is the simplest and one of the first methods used for summarization in the earlier days. The positional method comprises of extracting sentences from the first and last sentences.

In the same year (Luhn, 1958) proposed another extractive method based on TF-IDF[2]. Luhn removed stopwords and processed stemming on the input document. Analyzing such data found out that words which appear the most and those words that appear the least are not significant. A summary was produced by selecting sentences with the highest concentration of salient content terms, those in the interval between most and least common.

#### 2.2.1 Lead-3

Lead-3 is a method proposed by (Nallapati et al., 2016b), which extended the original positional method (discussed above), by simply producing the summary from the leading three sentences of the document. The method was proposed as a reaction to the Inverted Pyramid problem, discussed in the section 3.2.

[1] Cohesion is the grammatical and lexical linking within a text or sentence that holds a text together and gives it meaning.

[2] Term Frequency — Inverse Document Frequency

### 2.2.2 TextRank

TextRank (Mihalcea and Tarau, 2004) is a graph-based ranking algorithm based on PageRank.

PageRank (Page et al., 1999) is a ranking algorithm originally proposed to rank web pages in search engine results. Webpages are represented as nodes in the graph. The basic idea implemented by a graph-based ranking model is that of "voting" or "recommendation". When one vertex links to another one, it is basically casting a vote for that other vertex. The higher the number of votes that are cast for a vertex, the higher the importance of the vertex. Moreover, the importance of the vertex casting the vote determines how important the vote itself is. Formally speaking importance $S$ of webpage $V_i$ is defined as:

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j), \tag{1}$$

where $d$ is a damping factor that can be set between 0 and 1, which has the role of integrating into the model the probability of jumping from a given vertex to another random vertex in the graph. $In(V_i)$ represents inbounds links to the webpage $V_i$, $Out(V_i)$ outgoing links.

Starting from arbitrary values assigned to each node in the graph, the computation iterates until convergence below a given threshold is achieved. After running the algorithm, a score is associated with each vertex, which represents the "importance" of the vertex within the graph.

The original PageRank definition for graph-based ranking is assuming unweighted graphs. However, in TextRank the graphs are built from natural language texts and may include multiple or partial links between the units (vertices) that are extracted from text. It may be therefore useful to indicate and incorporate into the model the "strength" of the connection between two vertices $V_i$ and $V_j$ as a weight $w_{ij}$ added to the corresponding edge that connects the two vertices. Hence new webpage importance formula was introduced:

$$\begin{aligned} WS(V_i) &= (1-d) + \\ &+ d * \sum_{V_j \in In(V_i)} \frac{w_{ji}}{\sum_{V_k \in Out(V_j)} w_{jk}} WS(V_j), \end{aligned} \tag{2}$$

To apply TextRank for document summarization, sentence similarity must be defined. The similarity of two sentences can be determined simply as the number of common tokens $w_k$ between the lexical representations of the two sentences.

$$Sim(S_i, S_j) = \frac{|\{w_k | w_k \in S_i \wedge w_k \in S_j\}|}{log(|S_i|) + log(|S_j|)} \tag{3}$$

Other sentence similarity measures, such as string kernels, cosine similarity, longest common subsequence, etc. can be used. The summary is created from the $k$ sentences with the highest score.

### 2.2.3 BertSumExt

Extractive summarization systems create a summary by identifying (and subsequently concatenating) the most important sentences in a document. Neural models consider extractive summarization as a sentence classification problem: a neural encoder creates sentence representations and a classifier predicts which sentences should be selected as summaries. SummaRuNNer (Nallapati et al., 2016b) RNN based sequence model was one of the earliest neural approaches adopting an encoder based on Recurrent Neural Networks.

(Liu and Lapata, 2019) went further and fine-tuned BERT (Devlin et al., 2018) to achieve SOTA results across multiple datasets in both extractive and abstractive settings. Although BERT has been used to fine-tune various NLP tasks, its application to summarization is not as straightforward. Since BERT is trained as a masked-language model, the output vectors are grounded to tokens instead of sentences.

The authors proposed new architecture called BERTSUM displayed in Figure 1. BERTSUM extends BERT by inserting multiple [CLS] symbols to learn sentence representations and using interval segmentation embeddings (illustrated in red and green color) to distinguish multiple sentences. This way, document representations are learned hierarchically where lower Transformer layers represent adjacent sentences, while higher layers, in combination with self-attention, represent multi-sentence discourse.

Several inter-sentence Transformer layers (2 layers performed best in experiments mentioned in the paper) are stacked on top of BERT outputs, to capture document-level features for extracting summaries:

$$\tilde{h}^l = LN(h^{l-1} + MHAtt(h^{l-1})) \tag{4}$$

$$h^l = LN\tilde{h}^l + FFN(\tilde{h}^l)) \tag{5}$$

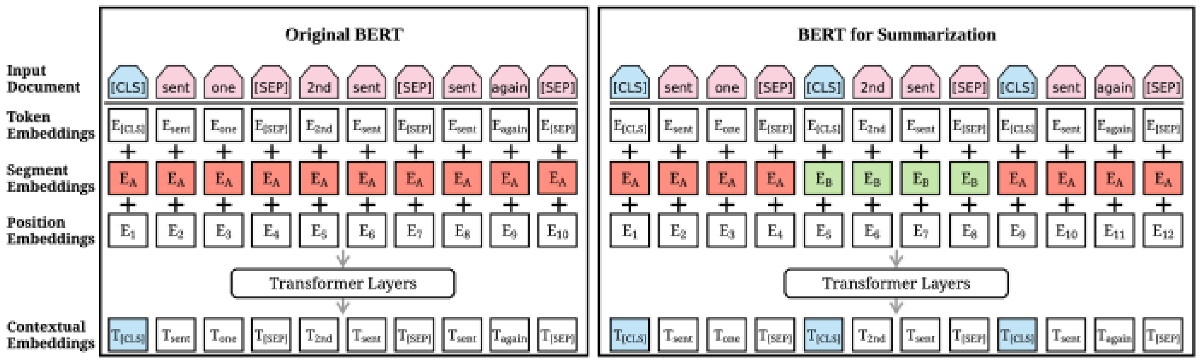


Figure 1: Architecture of the original BERT model (left) and BERTSUM (right). The sequence on top is the input document, followed by the summation of three kinds of embeddings for each token. The summed vectors are used as input embeddings to several bidirectional Transformer layers, generating contextual vectors for each token. The image and caption were adopted from the original paper (Liu and Lapata, 2019).

where $h^0 = PosEmb(T)$; $T$ denotes the sentence vectors output by BERTSUM, and function $PosEmb$ adds sinusoid positional embeddings to $T$, indicating the position of each sentence. The final output layer is a sigmoid classifier.

## 2.3 Abstractive

Abstractive summarization is an alternative technique that overcomes the drawbacks of extractive methods and is much nearer to the human approach to the problem.

Good quality abstractive summarization started to be possible with the progressive exploration of the huge neural networks in recent years. These models can model language and *understand* it. Models need to consume semantic information from the original document and produce the summary with completely new sentences. This process requires comprehension of the input text and the ability to generate text. Abstraction may transform the extracted content by paraphrasing sections of the source document, to condense a text more strongly than extraction. Such transformation, however, is computationally much more challenging than extraction.

Although nowadays abstractive summaries are pretty impressive, they still tend to generate false information called **hallucinations**. This could happen at either entity level (extra entities are generated) or entity relation level (context in which entities occur is incorrectly generated).

### 2.3.1 Bart

BART (Lewis et al., 2019), a denoising autoencoder for pretraining sequence-to-sequence models. BART is trained by corrupting text with an arbitrary noising function and learning a model to reconstruct the original text. It uses a standard Transformer-based neural machine translation architecture. Authors from the Facebook AI evaluated several noising approaches, finding the best performance by both randomly shuffling the order of the original sentences and using a novel in-filling scheme, where spans of text are replaced with a single mask token.

Because BART has an autoregressive decoder, it can be directly fine-tuned for abstractive summarization. In the summarization task information is copied from the input and is further manipulated, which is closely related to the denoising pre-training objective. The encoder input is the input sequence, and the decoder generates outputs autoregressively.

### 2.3.2 T5

Transfer learning, where a model is first pre-trained on a data-rich task before being fine-tuned on a downstream task, has emerged as a powerful technique in natural language processing (NLP). The effectiveness of transfer learning has given rise to a diversity of approaches, methodology, and practice.

(Raffel et al., 2019) proposed a unified framework, called T5[3], that converts all text-based language problems into a text-to-text format. Authors

[3]Text-to-Text Transfer Transformer

also introduced a new “Colossal Clean Crawled Corpus”[4]. The model achieved state-of-the-art results on many benchmarks covering summarization, question answering, text classification, and more.

Based on the original T5 model, Google has released some follow-up works:

- T5v1.1 - improved version of T5 with some architectural tweaks, only pre-trained on C4 excluding any supervised training. Therefore, this model has to be fine-tuned before it is useable on a downstream task, unlike the original T5 model.

- mT5 - a multilingual T5 model (101 languages)

- byT5 - a token-free model pre-trained on byte sequences rather than SentencePiece subword token sequences

## 2.4 Hybrids

Neural sequence-to-sequence models have provided a viable new approach for abstractive text summarization (meaning they are not restricted to simply selecting and rearranging passages from the original text). However, these models are liable to reproduce factual details inaccurately, and they tend to repeat themselves. On the other hand, extractive approaches may suffer from a lack of cohesion and the impossibility to compress data. Thus hybrid methods, which stand between those approaches were designed.

(See et al., 2017) proposed an architecture called Pointer-Generator that augments the standard sequence-to-sequence attentional model in two orthogonal ways. First, they use a hybrid pointer-generator network that can copy words from the source text via pointing, which aids accurate reproduction of information, while retaining the ability to produce novel words through the generator. Second, they use coverage to keep track of what has been summarized, which discourages repetition.

## 3 Datasets and evaluation

Currently used methods and architectures for understanding, modeling, and generating natural language require a huge amount of training data. Several publicly available NLP datasets were scrapped from the web in recent years. The creation of a dataset by scrapping is very cheap, however carries a risk of noisy, incomplete, or irrelevant data.

[4]https://github.com/google-research/text-to-text-transfer-transformer#c4

| | Train | Validation | Test |
|---|---|---|---|
| Pairs of data | 287 113 | 13 368 | 11 490 |
| Av. article length | 749 | 769 | 778 |
| Av. summary length | 53 | 61 | 58 |

Table 1: Basic statistics of the CNN/Daily Mail dataset. The source documents in the training set have 766 words spanning 29.74 sentences on average while the summaries consist of 53 words and 3.72 sentences.

### 3.1 CNN/Daily Mail Corpus

The CNN/Daily Mail Dataset (Hermann et al., 2015) is an English-language dataset containing just over 300 000 unique news articles as written by journalists at CNN and the Daily Mail. Table 1 shows basic statistics about the CNN/Daily Mail Dataset.

CNN/DM is very clean, has a low fraction of factual and data noise: there are no Incomplete/Irrelevant samples, and only 18% of samples belong to Entity Missing and Evidence Missing (55% of samples of another commonly used journalist dataset XSum belong to Entity Missing, Evidence Missing, or Incomplete classes). The dataset was designed to be abstractive in nature, and this is reflected in the distribution: 64% of samples belong to Paraphrase and Inference categories (Tejaswin et al., 2021).

CNN/DM has originally aimed to support supervised neural methodologies for machine-reading and question answering with a large amount of real natural language training data and released about 313k unique articles and nearly 1M Cloze style questions to go with the articles. Versions 2.0.0 and 3.0.0 changed the structure of the dataset to support summarization rather than question answering (Nallapati et al., 2016a).

The CNN articles were written between April 2007 and April 2015. The Daily Mail articles were written between June 2010 and April 2015. The dataset is curated from a a diverse range of news articles on topics like sports, health, business, lifestyle, travel, etc. (Bordia and Bowman, 2019) studied gender biases on several datasets and showed a good balanced ratio of female to male gender words of the CNN/DM.

|  |  |
|---|---|
| Dataset size | 230 843 |
| Average article length | 579.8 |
| Average summary length | 62.1 |
| Vocabulary size | 556 461 |

Table 2: Official statistics of the WikiHow dataset. Although authors claims size of dataset is over 200 000 of samples, 168 000 of articles are present in currently available version.

### 3.2 Wikihow

The majority of existing summarization datasets consist of news articles. These articles are written by journalists and follow the Inverted Pyramid style (Pöttker, 2003) to prioritize and structure a text by starting with mentioning the most important, interesting, or attention-grabbing elements of a story in the opening paragraphs and later adding details and any background information. This writing style might be the cause of relatively high lead-3 scores. Therefore, the WikiHow dataset (Koupaee and Wang, 2018), a large-scale text summarization dataset, was created. the WikiHow can be downloaded from the official GitHub repository[5]. It requires very complex summarization techniques to achieve good results.

The dataset consists of thousands of articles extracted from the WikiHow[6] knowledge base. Table 2 shows statistics about the WikiHow dataset. A WikiHow article typically consists of a headline, which is the question to be answered, followed by various steps, that explain how to solve the problem at hand. Each step consists of the main sentence, that could stand for itself, followed by some more sentences explaining this step in a more detailed way.

The summary is then generated by concatenating all main sentences, whereas the article is constructed by concatenating all explaining sentences. However, summaries generated by this way suffer from a lack of cohesion, which may favor extractive methods.

The articles span a wide range of topics written by people from different backgrounds and social groups (not only by journalists) and therefore the systems trained on WikiHow dataset may generalize better than those trained only on news.

[5]https://github.com/mahnazkoupaee/WikiHow-Dataset
[6]https://www.wikihow.com/

### 3.3 ROUGE - Evaluation metric

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation (Lin, 2004). It includes measures to automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans. The measures count the number of overlapping units such as n-gram, word sequences, and word pairs between the computer-generated summary to be evaluated and the ideal summaries created by humans.

Formally, ROUGE-N is an n-gram recall between a candidate summary and a set of reference summaries. ROUGE-N is computed as follows:

$$\text{ROUGE-N} = \\ = \frac{\sum_{S\in ref}\sum_{gram_n\in S} Count_{match}(gram_n)}{\sum_{S\in ref}\sum_{gram_n\in S} Count(gram_n)}, \tag{6}$$

where $n$ stands for the length of the n-gram $gram_n$, and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries.

The recall is great for ensuring our model is capturing all of the information contained in the reference. To ensure the model is not just pushing out a huge number of words to game the recall score precision metric is used. Precision is calculated in almost the same way, but rather than dividing by the reference n-gram count, the number of co-occurring n-grams is divided by the model n-gram count.

#### 3.3.1 ROUGE-L - Longest Common Subsequence

A sequence $Z = [z_1, z_2, ..., z_n]$ is a subsequence of another sequence $X = [x_1, x_2, ..., x_m]$, if there exists a strict increasing sequence $[i_1, i_2, ..., i_k]$ of indices of $X$ such that $\forall j \in \{1, 2, \ldots, k\} : x_{ij} = z_j$ (Cormen et al., 1989). Given two sequences $X$ and $Y$, the longest common subsequence (LCS) of $X$ and $Y$ is a common subsequence with maximum length.

### 3.4 Compression ratio

The compression ratio characterizes abstractness off the reference summarization. The compression ratio is defined as the ratio between the average length of the document and the average length of the summary. The higher the compression ratio, the more difficult the summarization task, as it needs to capture higher levels of abstraction and semantics. Table 3 shows the compression ratio for WikiHow and CNN/Daily Mail. The higher compression ratio of WikiHow shows the need for higher levels of abstraction, which may lead to worse results of the extractive methods.

| Dataset | **WikiHow** | **CNN/Daily Mail** |
|---|---|---|
| Document length | 100.68 | 118.73 |
| Reference summary length | 42.27 | 82.63 |
| Compression ratio | 2.38 | 1.44 |

Table 3: Reference summaries for WikiHow are only bold parts of articles and are slightly shorter in comparison with CNN/Daily Mail abstracts.

## 4 Results and Analysis

To get initial picture, how good methods mentioned in the section 2 are, they were evaluated on the part of "test" splits of the datasets mentioned above using ROUGE-1, Rouge-2 and Rouge-L metrics. Extractive methods were constrained to maximum length of 4 sentences and abstractive ones for 80 tokens. These numbers are based on the statistics of the used datasets.

Pre-trained models of the Bart[7], T5-small[8] and DistilBART[9][10] from the Hugging Face library were used for abstractive summarization. Extractive summarization based on the BERTSum was processed with fine-tuned[11] DistilBERT model (Sanh et al., 2019).

For the TextRanking algorithm 3 version to compute sentences similarities were used - original version from the paper, cosine similarity of the sentences Glove vectors[12] (mean vector of the words vectors) and top-ranked phrases based approach[13].

Figure 2 displays methods recalls and precisions on the CNN/Daily Mail dataset. Figure 3 displays methods recalls and precisions on the WikiHow dataset. Table 5 shows relative computation time per sample of used methods. There can be seen significant drop of performance of models fine-tunned on CNN/Daily Mail dataset. However Lead-N and TextRank scores are also much smaller due to the dataset specifics discussed in the section 3.2. Including reference summaries in the article texts led to the ROUGE-1 score $\approx 50\%$ for the TextRank based methods. For reasons of the poor performance of the T5-small model, WikiHow fine-tunned version was further examined. Comparison of methods scores is displayed in the table 4.

[7]https://huggingface.co/facebook/bart-large-cnn
[8]https://huggingface.co/t5-small
[9]https://huggingface.co/sshleifer/distilbart-cnn-12-6
[10]Knowledge distillation is the process of transferring knowledge from a large model to a smaller one. While large models have higher knowledge capacity than small models, this capacity might not be fully utilized. As smaller models are less expensive to evaluate, they can be deployed on less powerful hardware. In some specific cases may even obtain better results.
[11]https://github.com/chriskhanhtran/bert-extractive-summarization
[12]https://github.com/stanfordnlp/GloVe
[13]https://derwen.ai/docs/ptr/explain_summ/

| | T5-small | wikihow-T5-small |
|---|---|---|
| CNN/DM | 0.36 | 0.28 |
| WikiHow | 0.26 | 0.31 |

Table 4: Slightly better results where obtained by the fine-tunned version. Although there remains the problem with smaller model encoder size and to obtain better results, text should be split to chunks.

ROUGE and other automatic evaluation metrics are great tools, they come with some drawbacks mentioned below. Thus, brief qualitative analysis on several articles from the CNN/DM was made too.

- Automatic metrics only assess content selection and do not account for other quality aspects, such as fluency, grammaticality, coherence, etc.
- To assess content selection, they rely mostly on lexical overlap, although an abstractive summary could express they same content as a reference without any lexical overlap.
- To assess content selection, they rely mostly on lexical overlap, although an abstractive summary could express they same content as a reference without any lexical overlap.

### 4.1 Qualitative Analysis

## 5 Conclusion

This work compared several commonly used methods for single-document summarization. SOTA methods, especially abstractive ones, experienced enormous improvement over the last 5 years and nowadays provide superb outputs. However, there are the same lacks that need to be improved, I hope with the current process, we will have in several years almost flawless methods to even summarize long books or daily yield of news.

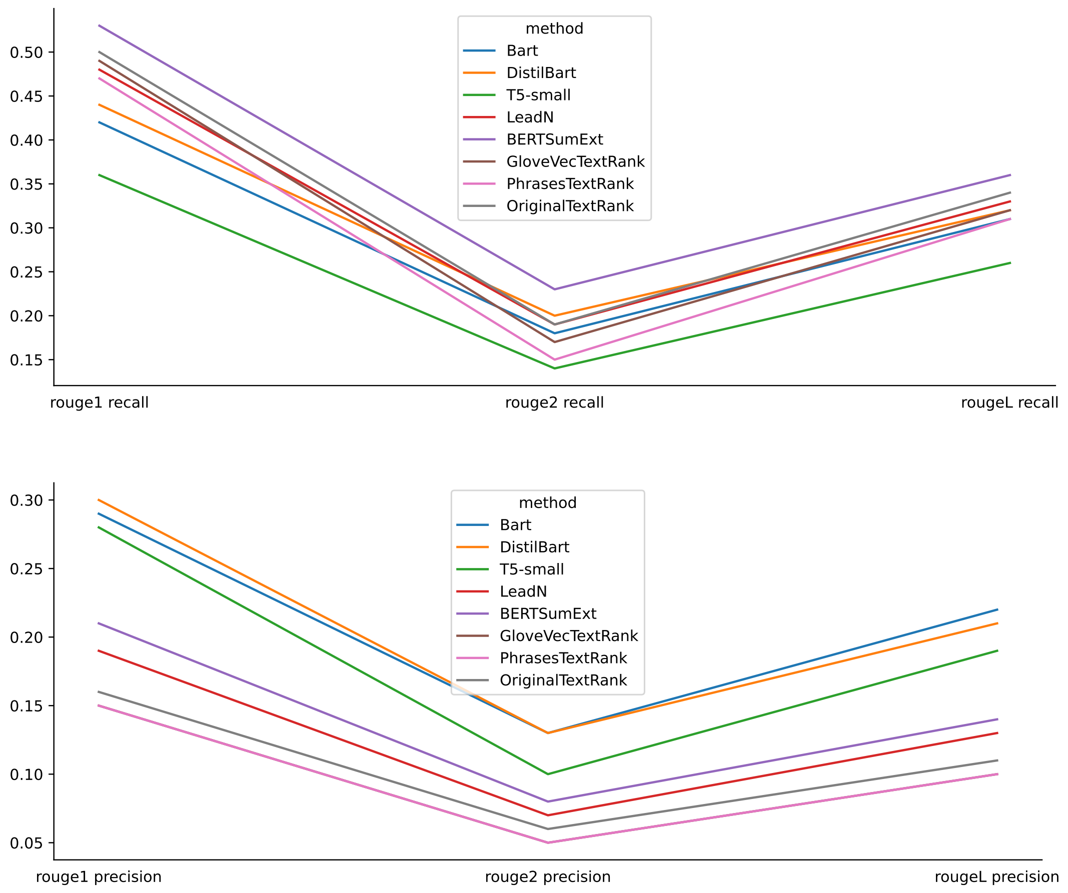
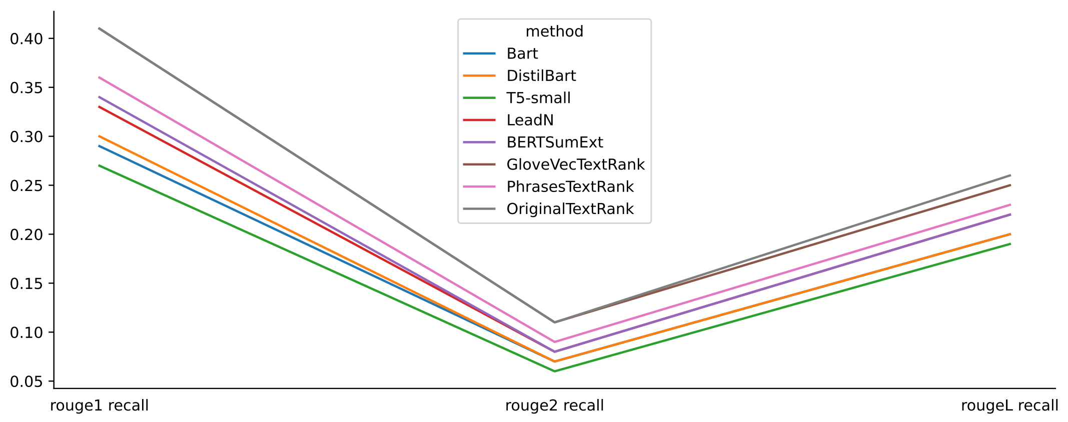


Figure 2: Evaluation of the methods on the CNN/Daily Mail dataset. Best ROUGE results were obtained by the BERTSumExt model. Better ROUGE-2 results for abstractive methods can be seen in the top picture. Abstractive models also obtain much higher ROUGE precission - ROUGE-1 $\approx 30\%$ in comparison to extractive methods ROUGE-1 precission that varies from 15% to 22 %. Distilled version of BART got slightly better results.



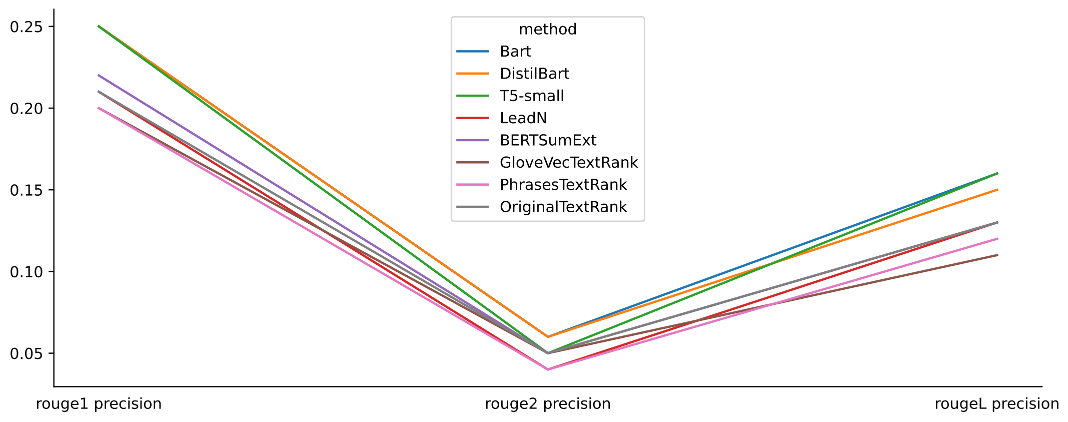


Figure 3: ROUGE scores of examined methods on the WikiHow dataset. Expected significantly lower performance of models fine-tunned on CNN/Daily Mail is present. DistilBART and BART generalize almost the same.

| Method | Time per sample |
|---|---|
| Bart | 6.79 |
| DistilBART | 4.45 |
| T5-small | 1.82 |
| BERTSumExt | 0.35 |
| GloveVecTextRank | 0.10 |
| PhrasesTextRank | 0.09 |
| OriginalTextRank | 0.09 |
| LeadN | 0.07 |

Table 5: Relative computation time per sample of examined methods. DistilBART got slightly better ROUGE results in 65% of BARTs time. BERTSumExt provides very good results in short time.

## References

P. B. Baxendale. 1958. Machine-made index for technical literature—an experiment. *IBM Journal of Research and Development*, 2(4):354–361.

Shikha Bordia and Samuel R. Bowman. 2019. Identifying and reducing gender bias in word-level language models. *CoRR*, abs/1904.03035.

Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. 1989. *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *CoRR*, abs/1810.09305.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *CoRR*, abs/1908.08345.

H. P. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016a. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2016b. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *CoRR*, abs/1611.04230.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120.

Horst Pöttker. 2003. News and its communicative quality: the inverted pyramid—when and why did it appear? *Journalism Studies*, 4(4):501–511.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

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

Priyam Tejaswin, Dhruv Naik, and Pengfei Liu. 2021. How well do you know your summarization datasets? *CoRR*, abs/2106.11388.