Hybrid Extractive/Abstractive Summarization Using Pre-Trained Sequence-to-Sequence Models

David Ferreira Coimbra

Instituto Superior Técnico - University of Lisbon
david.coimbra@tecnico.ulisboa.pt

Abstract. Typical document summarization methods can be either extractive, by selecting appropriate parts of the input text to include in the summary, or abstractive, by generating new text with basis on a meaningful representation of the source text. In both cases, the current state-of-the-art involves the use of pre-trained neural language models based on the Transformer architecture. Most of these approaches are unable to process input text beyond a limited small number of tokens. This paper advances a hybrid summarization approach based on a single T5 model, which first selects important sentences from the source text, and subsequently produces an abstractive summary from the selected sentences. In doing this, we reduce the overall computational requirements associated to the use of Transformer models, and mitigate the effects of their input size limitations, while ensuring a good performance. Through experiments with different datasets, we show that our method achieves comparable results to current state-of-the-art models, while maintaining relatively low computational requirements.

Keywords: Single Document Summarization · Natural Language Generation · Sequence-to-Sequence Neural Models · Transfer Learning

1 Introduction

In natural language processing (NLP), the goal of automatic summarization is to produce a document of shorter length than the source document while preserving its meaning. Currently, text summarization methods focus on two approaches: extractive summarization selects important fragments from the source document to be included in the final summary; while abstractive summarization generates new, paraphrased text that succinctly represents the knowledge in the source document. Most extractive summarization strategies reduce the problem to a classification task [13,18], where each text element (e.g. each sentence) is assigned a class label on the basis of belonging or not to the summary. Text elements are then ranked by the probability of the positive class.

Abstractive summarization has seen rapid development with the emergence of sequence-to-sequence models: encoder-decoder architectures which convert the source text into an internal representation and subsequently decode this representation into new text. Implementations leveraging attention-based feed-forward networks [22] train an encoder and a generation model jointly on a single summarization task. Pointer-generator networks [24] leverage the attention mechanism to copy words from the source text while maintaining paraphrasing and generalization abilities, improving summary coherence. Additionally, these networks employ a coverage mechanism to prevent repetition. Hybrid models [12,26] employ extractive and abstractive modules, separating the summarization process into distinct stages of content selection and paraphrasing.

Abstractive summarization is, conceptually, a significantly harder problem than extractive, as abstractive models struggle with coherence, repetition, and accurate representation of factual knowledge. For these reasons, most studies focus on extractive approaches [8]. However, while extractive summarization provides more coherent summaries, it lacks sophisticated abilities that are characteristic of high-quality summarization; such as paraphrasing, generalization, and incorporation of knowledge external to the environment under study [24]. Currently, state-of-the-art performance in the majority of NLP tasks is achieved by applications of the Transformer architecture [29], which are pre-trained in a language modeling objective and subsequently fine-tuned on a downstream task. This pre-training regime allows these models to incorporate large amounts of knowledge that is external to the dataset under study. Bidirectional encoding models such as BERT [6] have proven highly effective in a variety of classification tasks; while sequence-to-sequence models such as BART [10], T5 [21] and PEGASUS [34] have built upon the Transformer architecture to introduce advancements in text generation tasks. Despite these advantages, these pre-trained models are unable to process text beyond a maximum amount of 512 tokens, making them unsuitable for summarization of long documents. More recent models such as Longformer [1] address this shortcoming by implementing sliding windows, but the large computational requirements inherent to Transformer models make it prohibitive to carry out a successful fine-tuning routine without using less expressive models or destroying information [25].

To mitigate the effects of the token limit and the computational requirements of large Transformer models, we present a hybrid fine-tuning routine for the T5 model that iteratively learns to select important sentences from the source document and produces an abstractive summary from the selected sentences. Our intuition is that by informing the model of the parts of the document that are most important, it will preserve that information in the final abstractive summary. By leveraging external knowledge existent in the model from pre-training, we hypothesize that this approach will lead to comparably high results with much lower computational requirements, as it will not be necessary to encode the entire source document at once. To assess the applicability of this method in different data distributions, we train and evaluate the model on the TripAdvisor and arXiv datasets for summarization, characterized by short and long documents respectively. We show that performing a pre-processing step with an extractive model results in significant benefits over a standalone abstractive approach.

The remainder of this document is organized as follows: Section 2 presents related work, while Section 3 describes our hybrid method. Section 4 describes our experiments, presenting the datasets and evaluation metrics, the model training setup, and finally discussing the obtained results. Finally, Section 5 summarizes our conclusions, and presents possible directions for future work.

2 Related Work

Tarnpradab et al., 2018 [27] were the first to employ an approach based on deep neural networks for the extractive summarization of online discussions. The authors employ a hierarchical attention network (HAN) [33], which is trained to learn sentence representations by attending to important words and subsequently learn representations for the entire discussion in a hierarchical manner. Sentences are encoded by replacing each word with a pre-trained embedding generated through Word2Vec and feeding them to a bidirectional LSTM. Building upon this work, Magooda and Marcjan, 2020 [15] hypothesize that attending to the initial sentences of an online discussion incorporates the model with topical knowledge that is concentrated in the first post. Thus, the authors employ additional attention vectors by pairing every sentence of the document with each sentence of the first post of the document. Bidirectional attention is computed from a similarity matrix between each sentence pair. Each row of the similarity matrix, after softmax normalization, represents document-to-first-post attention while each column represents first-post-to-document attention. This approach is applied to two models for extractive summarization: SummaRuNNer [17] (an auto-regressive model) and SiATL [3] (a non-auto-regressive model). The best results were achieved with SiATL, as non-auto-regressive models tend to perform well in sentence classification tasks [32].

The task of summarizing long scientific texts using deep neural models was first explored by Cohan et al., 2018 [4], who employed an abstractive discourse-aware model based on an encoder-decoder architecture. This model takes advantage of the general structure of a scientific article, attending to different discourse sections. The encoder is a hierarchical RNN that learns representations for each discourse section and subsequently for the entire document. The decoder is enhanced with special attention vectors that attend to the relevant discourse section. To address the problem of unknown token prediction, the model includes an additional binary classifier that decides whether the next word should be generated from the vocabulary or copied from the source document. Xiao and Carenini, 2019 [32] expand this approach to extractive summarization by combining local (section-level) and global (document-level) context. At the document-level, a bidirectional GRU produces sentence representations by concatenating the backward forward hidden states for each sentence. Document representations act as global context, computed by concatenating the final state of the forward and backward GRU. Local context is captured through the LSTM-minus method, where each section is represented as the subtraction between the hidden states of the start and the end of that section. Finally, Tretyak and Stepanov, 2020 [28] combine extractive and abstractive approaches. The authors' proposed model consists of two components: a BERT-based classifier that selects sentences from the source document which should be included in the summary, and an abstractive language model that conditionally generates a summary from the source text conditioned with the abstractive summary. The experiments for the latter component are carried out on GPT-2 [20] and BART [10] models.

3 The Proposed Hybrid Approach

We use T5 (Text-To-Text Transfer Transformer) [21] for all our experiments. We chose this model for its applicability to all tasks in NLP, its unified interface for training, and for the low computational requirements of its smaller version. T5 treats every task in NLP as a text-to-text problem, i.e., taking text as input and returning text as output. This unified framework allows us to directly apply the same model to both extractive and abstractive summarization, retaining the same hyperparameters and training procedure. We use the smallest version of the model, which spans a total of 60 million parameters, through Hugging Face's Transformers library [31]. T5's tokenizer is based on the SentencePiece tokenizer [9]. The pre-trained model uses a fixed vocabulary of 32128 word pieces; as a pre-processing step, the tokenizer splits out-of-vocabulary words into word pieces that are in the vocabulary, converting these word pieces into vocabulary indices for the model to take as input. Given an input sequence $\boldsymbol{s} \in \mathcal{V}^l$ of tokens that are part of vocabulary $\mathcal{V}$ and of length l, the T5 model generates a sequence $\boldsymbol{o}$ of tokens of user-defined length m, i.e., $\boldsymbol{o} \in \mathcal{V}^m$. The model's output is a matrix $\boldsymbol{R} \in \mathcal{V}^m \times \mathcal{V}^n$, where n is the size of $\mathcal{V}$ and every entry $\boldsymbol{R}_{ij}$ is the logit (a "prediction score") that the i-th output token is equal to the j-th vocabulary entry. The fine-tuning process for T5 involves prepending a special prefix to the input sequence, unique for each downstream task. This allows T5 to adjust itself for any particular task, and for the same model to be trained simultaneously in different tasks, as we do in this work.

3.1 Extractive Summarization

To produce an extractive summary, we consider each sentence as a standalone sample, to be classified as belonging or not to the summary. Let $\boldsymbol{s} = [s_1, \ldots, s_N]$ be the sentences in a document of N sentences and $\boldsymbol{t} = [t_1, \ldots, t_N]$ to be the binary labels, where "true" indicates the sentence is in the summary and "false" otherwise. The task of extractive summarization finds the most probable sequence of labels, given the source sentences:

$$\arg\max_{\boldsymbol{t} \in \mathcal{T}} p(\boldsymbol{t}|\boldsymbol{s}) \tag{1}$$

where $\mathcal{T}$ is the set of all possible label sequences. As all labeling decisions are independent in this work, $p(\boldsymbol{t}|\boldsymbol{s}) = \prod_{i=1}^{N} p(t_i|\boldsymbol{s})$. This contrasts with auto-regressive models, in which previous decisions made by the model affect its future decisions [17]. To make a prediction, we provide the following sequence:

`important sentence:` s_i `</s>`

where $s_i \in \boldsymbol{s}$ and `</s>` is the end-of-sequence token. When generating a prediction, we follow Nogueira et. al, 2020 [19] by applying a ranking approach based on the predictions of the target words "true" and "false". In particular, we evaluate the pair of logit values $t = \boldsymbol{R}_{0,'\text{true}'}; f = \boldsymbol{R}_{0,'\text{false}'}$. We then compute the activation $y = \sigma(t - f)$, where σ denotes the sigmoid function. On inference, we rank each sentence in the document by the corresponding activation value, and limit the maximum number of sentences to the average number of sentences in the reference summaries.

3.2 Abstractive Summarization

Let $\mathcal{X}$ be the set of all possible input sequences and $\mathcal{Y}_N$ be all possible sequences of user-defined length N. The task of abstractive summarization finds the optimal sequence $\boldsymbol{y} \in \mathcal{Y}_N$ under a scoring function $s : \mathcal{X} \times \mathcal{Y}_N \mapsto \mathbb{R}$:

$$\arg\max_{y \in \mathcal{Y}_N} s(\boldsymbol{x}, \boldsymbol{y}) \tag{2}$$

In the previous expression, $\boldsymbol{x} \in \mathcal{X}$. We carry out two distinct experiments for abstractive summarization:

– We take the source document as-is and truncate it to the first 512 word-pieces, which the model takes as input.
– Following Subramanian et al., 2019 [26] and Tretyak and Stepanov, 2020 [28], we perform an extractive step before training, to reduce the source document to its important sentences.

Our input for this task is the following:

`summarize:` d `</s>`

where d is the document to summarize. We train the abstractive model on the reference summaries for each dataset. Training is performed in "teacher-forcing" fashion [30], where the input and target sequences are directly fed to the model. We use greedy decoding, and decode until an end-of-sequence token is emitted or we reach a maximum length equal to the average length of the reference summaries. The remaining parameters keep the library defaults.

3.3 Combining Extractive and Abstractive Summarization

To build our hybrid method for summarization, we iterate across three steps:

1. We perform one training epoch in the extractive context.
2. Using the model trained in the aforementioned step, we reduce the training set to a set of extractive summaries.

Table 1: Metrics for the TripAdvisor and arXiv datasets. Note that the all entries, except for documents and vocabulary size, are average values.

Dataset	Docs.	Vocab. size	Sentences /doc	Words /doc	Words /sentence	Sentences /summary	Words /summary	Words /summary sentence
TripAdvisor	700	23414	56	974	16	11	217	18
arXiv	215K	200K	207	4938	29	9	220	34

3. We perform one epoch in the abstractive context, taking as input the extractive summaries generated in the aforementioned step.

The previous process is repeated for the defined number of epochs. The resulting model can be used in both extractive and abstractive summarization tasks, and is evaluated in both these contexts. Note that we avoid generating the "true" and "false" tokens during the abstractive context, by instructing the decoder to ignore them.

4 Experimental Evaluation

The main focus of this work is to assess the adequacy of our method to the domain of online discussions, as texts in this domain are usually short and lend themselves well to a summarization task; where important information pertains to solutions to a presented problem [2]. In addition, we evaluate the generalization ability of our method by applying it to the larger, more variable domain of scientific documents. These are often much longer than online discussions, but follow a standard discourse structure (introduction, methodology, results, and conclusion). Scientific documents often include an abstract, which is meant to serve as an appropriate summary of the motivations and conclusions presented in the document [4]. Statistics for both datasets are presented in table 1.

The TripAdvisor forum discussions dataset [2,27] consists of 700 TripAdvisor discussions each annotated with three summaries, one extractive and two abstractive. Following Magooda and Marcjan, 2020 [15], 100 of these discussions correspond to the test set, 100 make up the validation set, and the remaining 500 discussions correspond to the training set. Pre-processing of the dataset was limited to concatenating the text of all posts in each discussion to obtain a single document. Tarnpradab et al., 2018 [27] obtained the extractive summaries through a greedy method: by iteratively adding one sentence at a time to the summary and checking if it has raised the ROUGE score.

To evaluate our approach with a larger, more variable data distribution, characterized by longer texts with higher variance, we apply the approaches described in the previous section to the arXiv dataset [4], which collects approximately 215K documents published in the scientific repository `arXiv.org`. Approximately 5% of the dataset is retained as validation data and another 5% is used as test data; the rest is used for training. This dataset has been preprocessed to remove figures and tables, and to normalize math formulas and citations with special tokens. Documents that were too short or too long, or that did not include an abstract, were excluded. Each document has been properly annotated with section indicators to easily extract specific sections from the document body (e.g. introduction or conclusion); however, we always consider the entirety of the document. The summarization task for this dataset is to predict the abstract based on the document body. For the extractive summaries, we leveraged the work of Xiao and Carenini, 2019 [32], which have annotated each sentence in each document with an appropriate label, using the same method used for the TripAdvisor dataset, for a total of around 42 million sentences on the training set. We extract each sentence and its respective label to a separate file.

4.1 Evaluation Metrics

Most automatic evaluation of natural language generation methods for text summarization involve measuring the overlap of n-grams between the generated summary and a reference summary. This is a straightforward strategy that allows for automatic evaluation of summaries in a convenient fashion. However, there is a set of limitations adjacent to using this approach as a means to claim state-of-the-art [7,8]:

1. Overlap-measuring strategies assess only content selection, and as such provide only a measure of adequacy for the summary; they do not account for other quality aspects, such as fluency and coherence.
2. In abstractive summarization, there are many ways to represent the same factual knowledge as in the reference summary without using the same n-grams, making a metric based on this principle unsuitable for this task.
3. Summarization is a subjective task, and subsequently the metrics are designed to account for multiple references per input. However, most datasets, such as the arXiv dataset used in this study, provide only one reference.
4. As noted by [5], the high correlation with human judgments shown by the ROUGE authors consist of news data, which is intrinsically very different than other summarization tasks, such as the summarization of online discussions or scientific papers.

For these reasons, we evaluate our findings using two metrics in a complementary fashion: ROUGE measures overlapping units between the generated summary and the reference summaries, such as n-grams, word sequences, and word pairs; while BERTScore computes a similarity score between the embeddings of the two sequences' tokens. Both metrics report precision, recall, and F1-score values. We use the latter to determine summary adequacy, as it provides an informative result incorporating both accuracy and recall. We describe both metrics in this section.

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [11] is the standard set of metrics used for evaluating automatic text summarization. It measures overlap between generated text and a set of references. A ROUGE score describes the adequacy of the summary, i.e., the amount of important information represented in the generated summary.

ROUGE-N measures n-gram overlap. For comparison with previous works, we measure ROUGE-1 (unigram overlap) and ROUGE-2 (bigram overlap). ROUGE-N is designed to favour candidate summaries that contain words shared by more references, giving more weight to a summary that aligns with consensus.

ROUGE-L takes into account the longest common subsequence between the system and the generated summaries. The longer the LCS of two summary sentences, the higher the ROUGE-L score of the candidate summary.

BERTScore [35] addresses common pitfalls in overlap-based metrics, being able to consider paraphrases. Instead of considering exact overlap between text elements, it computes the cosine similarity between the BERT embeddings of the generated sequence and the target sequence; where each token in the candidate sequence is matched to the most similar token in the reference sequences. Token alignments are computed greedily, maximizing the cosine similarity between contextualized token embeddings. In our experiments, we compute the BERTScore using the `distilbert-base-uncased` pre-trained weights. Additionally, because of the learned geometry of contextual embeddings, the numbers provided by BERTScore often fall into a limited range. This characteristic does not impact BERTScore's evaluation capabilities, but it makes the actual score less readable. For this reason, the score is rescaled linearly with respect to an empirical lower bound b as a baseline. b is computed by averaging BERTScore computed on one million random candidate-reference sentence pairs sampled from Common Crawl monolingual datasets.

4.2 Experimental Setup

All experiments were run under PyTorch 1.6.0 and Hugging Face's Transformers 3.3.0, on an Nvidia Tesla T4 GPU supplied by Google Cloud Platform's Compute Engine. To minimize the amount of boilerplate code, we wrote our training procedures using the PyTorch Lightning library, on version 0.9.0. All experiments made use of PyTorch's native automatic mixed precision casting to reduce the memory requirements of the training procedure. We follow the recommendations by Mosbach et. al, 2020 [16] for fine-tuning BERT models by using a batch size of 16, a learning rate of 2e-5 and a schedule that increases the learning rate linearly for the first 10% of steps and linearly decreases it afterwards. We set the weight decay value to 0.01 and clip gradient norms to 1.0. We use the AdamW optimizer [14]. As this is a general baseline for fine-tuning BERT, our intuition is that it is adequate for T5, which is Transformer-based as well. Note that for the abstractive context, it was necessary to use a batch size of 8 and accumulate gradients each 2 steps to approximate a batch size of 16. For the TripAdvisor dataset, we train for 20 epochs and evaluate on the epoch that returns the highest ROUGE score on the validation set. For the arXiv dataset, we train for only one epoch, due to time and hardware constraints. Note that in our iterative hybrid experiment, we instantiate two AdamW optimizers, each for the extractive and abstractive steps.

Table 2: F1-scores for the TripAdvisor dataset. Our contributions are in bold.

Method	Params.	Type	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
Tarnpradab et al [27]		Ext.	37.60	14.40	33.80	–
Magooda and Marcjan [15]		Ext.	46.50	28.53	44.65	–
DistilBERT	66M	Ext.	38.96	15.85	24.77	39.11
T5 Classifier	60M	Ext.	35.87	13.54	22.56	37.02
T5 Hybrid	60M	Ext.	36.38	14.14	23.21	37.20
T5 Summarizer	60M	Abs.	35.47	12.65	21.10	38.16
T5 After Extractive	120M	Abs.	37.05	14.38	21.72	39.15
T5 Hybrid	60M	Abs.	36.01	13.32	21.21	38.72

Table 3: F1-scores for the arXiv dataset. Our contibutions are in bold.

Method	Params.	Type	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore
Cohan et al. [4]		Ext.	35.80	11.05	31.80	–
Xiao and Carenini [32]		Ext.	43.62	17.36	29.14	–
Tretyak and Stepanov [28]	110M	Ext.	45.40	20.90	33.70	–
T5 Classifier	60M	Ext.	40.53	12.66	22.13	43.14
T5 Hybrid	60M	Ext.	40.41	12.24	21.62	42.92
Tretyak and Stepanov [28]	249M	Abs.	45.30	25.10	36.20	–
T5 Summarizer	60M	Abs.	31.50	8.47	19.82	31.27
T5 After Extractive	120M	Abs.	38.52	13.23	23.84	39.61
T5 Hybrid	60M	Abs.	34.15	10.47	22.31	35.72

4.3 Experimental Results

We present the results of all our experiments in table 2 for the TripAdvisor dataset and in table 3 for the arXiv dataset. As a baseline, we performed an experiment within an extractive context using the DistilBERT model [23], a faster and smaller version of BERT. For both models, extractive methods remain the most effective. Among abstractive summarization, performing an extractive step before training in the abstractive context exceeds all scores relative to the baseline abstractive method, showing that training on an extracted summary reduces the amount of unimportant information that the model needs to consider. The implementation of our hybrid method does not seem to yield significantly better results than the standalone models in both summarization tasks. However, the abstractive model trained after an extractive step remains the most significant improvement over standalone T5. Therefore, while our proposed model has not proven to be beneficial relatively to standalone methods, our experiments show that abstractive summarization can be greatly improved by employing an extractive pre-processing step beforehand.

5 Conclusions and Future Work

We presented a hybrid method for text summarization that iteratively trains the T5 model in extractive and abstractive contexts, by considering each sentence as an isolated sample and subsequently reducing the source documents to the important sentences. We show that our approach leads to results comparable to baseline methods, at a much lower computational requirement than taking the entirety of a source document as model input. From our hybrid experiments, we concluded that while the best results are achieved with extractive summarization, abstractive summarization models perform better after reducing each document to its most important sentences via an extractive step beforehand.

In the future, we aim to explore some simple enhancements to our training routine to improve the effectiveness of the hybrid model, such as paying attention to certain parts of each document (e.g. the beginning of an online discussion [15] or the concluding statements of a scientific paper), or experimenting with transformations of the data (e.g. instead of reducing a document to its most important sentences, simply condition the model with the extractive summary by concatenating it with the source document [26]). Finally, we would like to refine our current approach to larger datasets, using increased computational resources, that might allow our method to handle data-rich environments better as our constraints to small datasets and models, from a conceptual standpoint, are only based on available resources. In particular:

- Use larger versions of the T5 model, with more parameters, which we speculate will lead to better results on larger, more variable datasets;
- For the arXiv dataset, combine the previous point within a longer training regime, leveraging the validation set; potentially combine this approach with larger batch sizes and single-precision training instead of mixed precision.
- Experiment with other state-of-the-art Transformer-based models with conditional generation capabilities, such as BART and PEGASUS.
- Explore applications of our method combined with models designed to work with longer input sequences, such as Longformer, for the summarization of longer documents such as the ones in the arXiv dataset.

References

1. Beltagy, I., Peters, M.E., Cohan, A.: Longformer: The long-document Transformer. arXiv preprint arXiv:2004.05150 (2020)
2. Bhatia, S., Biyani, P., Mitra, P.: Summarizing online forum discussions – can dialog acts of individual messages help? In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (2014)
3. Chronopoulou, A., Baziotis, C., Potamianos, A.: An embarrassingly simple approach for transfer learning from pretrained language models. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (2019)

4. Cohan, A., Dernoncourt, F., Kim, D.S., Bui, T., Kim, S., Chang, W., Goharian, N.: A discourse-aware attention model for abstractive summarization of long documents. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (2018)
5. Cohan, A., Goharian, N.: Revisiting summarization evaluation for scientific articles. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16) (2016)
6. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional Transformers for language understanding. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (2019)
7. Fabbri, A.R., Kryściński, W., McCann, B., Xiong, C., Socher, R., Radev, D.: SummEval: Re-evaluating summarization evaluation. arXiv preprint arXiv:2007.12626 (2020)
8. Kryscinski, W., Keskar, N.S., McCann, B., Xiong, C., Socher, R.: Neural text summarization: A critical evaluation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (2019)
9. Kudo, T., Richardson, J.: SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing: System Demonstrations (2018)
10. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (2020)
11. Lin, C.Y.: ROUGE: A package for automatic evaluation of summaries. In: Text Summarization Branches Out (2004)
12. Liu, P.J., Saleh, M., Pot, E., Goodrich, B., Sepassi, R., Kaiser, L., Shazeer, N.: Generating Wikipedia by summarizing long sequences. arXiv preprint arXiv:1801.10198 (2018)
13. Liu, Y., Lapata, M.: Text summarization with pretrained encoders. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (2019)
14. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
15. Magooda, A., Marcjan, C.: Attend to the beginning: A study on using bidirectional attention for extractive summarization. arXiv preprint arXiv:2002.03405 (2020)
16. Mosbach, M., Andriushchenko, M., Klakow, D.: On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. arXiv preprint arXiv:2006.04884 (2020)
17. Nallapati, R., Zhai, F., Zhou, B.: SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In: Proceedings of the AAAI Conference on Artificial Intelligence (2017)
18. Nallapati, R., Zhou, B., dos Santos, C., Gulçehre, Ç., Xiang, B.: Abstractive text summarization using sequence-to-sequence RNNs and beyond. In: Proceedings of The SIGNLL Conference on Computational Natural Language Learning (2016)
19. Nogueira, R., Jiang, Z., Lin, J.: Document ranking with a pretrained sequence-to-sequence model. arXiv preprint arXiv:2003.06713 (2020)
20. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI Blog (2019)

21. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text Transformer. Journal of Machine Learning Research (2020)
22. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (2015)
23. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019)
24. See, A., Liu, P.J., Manning, C.D.: Get to the point: Summarization with pointer-generator networks. In: Proceedings of the Annual Meeting of the Association for Computational Linguistics (2017)
25. Sohoni, N.S., Aberger, C.R., Leszczynski, M., Zhang, J., Ré, C.: Low-memory neural network training: A technical report. arXiv preprint arXiv:1904.10631 (2019)
26. Subramanian, S., Li, R., Pilault, J., Pal, C.: On extractive and abstractive neural document summarization with Transformer language models. arXiv preprint arXiv:1909.03186 (2019)
27. Tarnpradab, S., Liu, F., Hua, K.: Toward extractive summarization of online forum discussions via hierarchical attention networks. In: Proceedings of the International Florida Artificial Intelligence Research Society Conference (2017)
28. Tretyak, V., Stepanov, D.: Combination of abstractive and extractive approaches for summarization of long scientific texts. arXiv preprint arXiv:2006.05354 (2020)
29. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in Neural Information Processing Systems (2017)
30. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation (1989)
31. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al.: HuggingFace's Transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771 (2019)
32. Xiao, W., Carenini, G.: Extractive summarization of long documents by combining global and local context. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (2019)
33. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics (2016)
34. Zhang, J., Zhao, Y., Saleh, M., Liu, P.J.: PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. arXiv preprint arXiv:1912.08777 (2019)
35. Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: BERTScore: Evaluating text generation with BERT. In: Proceedings of the International Conference on Learning Representations (2020)