Reproducing Abstractive Text Summarization with Pretrained Encoders

JAN HEINRICH REIMER, Martin Luther University Halle-Wittenberg, Germany

Large pretrained transformer encoders like Bert [Dev+19] have successfully been used for various tasks from the field of natural language processing [LL19]. Liu and Lapata build upon the pretrained Bert model to tackle extractive and abstractive text summarization [LL19]. Their deep neural abstractive summarization model outmatches many previously state-of-the-art summarizers in informativeness and fluency measured using ROUGE metrics [LL19; Lin04]. Since their publication, other fine-tuning models were introduced that surpass their performance [Agh+20; SAR20]. In this report, we reproduce the abstractive summarization experiments of Liu and Lapata using the Flux [Inn+18] framework by re-implementing their BertSumAbs model. We highlight differences and difficulties in experimental replication to build a better understanding of abstractive summarization using pretrained encoders.

1 INTRODUCTION

The Web enables access to huge amounts of textual data, like news articles. However, most web documents are unstructured and thus difficult to analyze. For instance, it is hard to follow current news events, if we would have to read each article entirely. Text summarization in general can be used to condense larger texts to its essential contents [Tor14, p. xix]. Often software is used for automatic text summarization and in recent years automatic text summarization has become an important problem in natural language processing for information retrieval [Tor14, p. xxi], often tackled by deep neural networks [Cel+18; LL19; Nal+16; PXS18; SLM17]. The task of text summarization can be divided into two main approaces: extractive and abstractive summarization. In extractive text summarization, the task is to extract most important sentences from the source text. In abstractive text summarization new text is generated which does not necessarily have to match sequences of the source [Tor14, p. 28] nor its vocabulary [Nal+16]. Particularily, abstractive summaries generated automatically aim to be more comparable to summaries prouced by human [Tor14, p. 220].

Previous neural approaches to abstractive summarization are based on sequence-to-sequence modelling [Nal+16], pointing to the source text [SLM17], or communicating agents [Cel+18]. Though, the limited amount of training data in benchmark datasets suited for abstractive summarization makes it difficult to train deeper neural networks. For example, the articles from the CNN/Daily Mail datasets contain only 242M words of text, that could be used for training an abstraction model [Her+15]. With the availability of large pretrained language models like ELMo [Pet+18] and Bert [Dev+19] it is often more efficient to fine-tune a pretrained language model, that already encodes semantics of a language [LL19]. The Bert model, for instance, was trained on 3300M words [Dev+19], more than 10 times the training data available from the CNN/Daily Mail datasets. Apart from

the advantages of higher amounts of training data, a pretrained language model could also potentially reduce learning cost, as models building upon a pretrained model only have to be fine-tuned, not trained from scratch [Dev+19].

Liu and Lapata propose two new approaches to text summarization that leverage a pretrained Bert model: a sentence classification architecture for extractive summarization (BertSumExt) and an encoder-deoder framework for abstractive summarization (BertSumAbs) [LL19]. In both tasks, the source text is encoded with pretrained Bert transformers. Then for extractive summarization, sentence sequences from the Bert encoder are again transformed and then classified whether they should be included in the summary. For abstractive summarization, Bert output embeddings are decoded using a transformer decoder and a linear layer is used to map embeddings back to the input vocabulary [LL19]. In both cases, the Bert encoder is jointly fine-tuned with input emeddings, classfier and/or decoder layers. Notably, the abstractive summarization model, which we reproduce in this report, follows an architecture very similar to neural machine translation and a neural machine translation toolkit is used in many parts of the BertSumAbs implementation [LL19; Kle+17].

From the models proposed by Liu and Lapata, we choose the BertSumAbs abstractive summarization model for our replication due to its promising trade-off between model complexity and performance in terms of ROUGE metrics on the CNN/Daily Mail datasets [LL19; Her+15]. Compared to previous neural approaches [SLM17; PXS18], it does not need copy nor coverage [LL19] Furthermore, the abstractive model also allows for more variation of the model's hyperparameters, e.g. changing the depth or width of the decoder's transformer layers. Even though the extractive model BertSumExt outperforms previous approaches more distinctly, it requires more complex preprocessing of input articles and the model changes aspects of Bert's internals [LL19], requiring a separate, more detailed replication study.

Liu and Lapata claim that the BertSumAbs model outperforms many previous state-of-the-art summarizers in informativeness and fluency, measured using the ROUGE-1, ROUGE-2, and ROUGE-L metrics [LL19; Lin04]. We expect similar scores with our re-implemented model, but due to the lack of hardware resources for training such a big neural model we are unable to compare final ROUGE scores. With our more idiomatic re-implementation we facilitate more in-depth analyzes of the BertSumAbs and TransformerAbs models by Liu and Lapata. The technical difficulties we faced in training the BertSumAbs model questions the usefullness of very large, pretrained encoder models like Bert for abstractive text summarization, at least on a smaller budget. It remains an open question whether smaller variants like Bert-Tiny could achieve comparable ROUGE scores for abstractive summarization [Tur+19].

2 RELATED WORK

In the reproduced article, Liu and Lapata use the encoder from the pretrained Bert language model to build an absractive text summarization model [LL19; Dev+19]. While neural architectures have already been used previously for abstractive

text summarization [Nal+16; SLM17; PXS18], it has been difficult to generate fluent yet informative summaries as the limited data available from benchmark datasets is often unsufficient for training deep neural networks [Nal+16]. Though, these deep networks are required for modelling linguistics needed to abstract text. Particularly, common benchmark datasets like the CNN/Daily Mail datasets for supervised learning contain far less data than the amount available for unsupervised pretraining of language models [Her+15; Dev+19]. By fine-tuning a transformer encoder-decoder architecture with pretrained encoders [Vas+17; Dev+19], Liu and Lapata leverage broader linguistic knowledge from the pretrained language model for generating better summaries [LL19]. As predicted in their article, now more advanced language models, e.g., UniLM, are able to generate even better abstractive summaries [Don+19].

2.1 Abstractive Summarization

Abstractive text summarization is the process of generating a more compact representation of a text sequence's essential content while not having to copy from the source exactly [Tor14, p. 28]. Instead, the generated summary can have different style or include other vocabulary [Nal+16].

Nallapati et al. were among the first to apply neural methods to abstractive summary generation in a sequence-to-sequence setting. They use an encoder-decoder recurrent neural network model that was originally developed for neural machine translation [BCB15] but improve by letting the decoder decide between pointing to a token from the source text for copying or generating a new token [Nal+16]. Even though both machine translation and abstractive summarization can be modelled as a sequence-to-sequence task, they are different in length of generated summaries/translations and their lossiness [Nal+16]. The pointer/generator approach [Nal+16] was later improved on by considering coverage, as See, Liu, and Manning found that unconstrained sequence-to-sequence models tend to produce repetitive, unnatural summaries [SLM17; PXS18]. Paulus, Xiong, and Socher take into account attention over input and generated output to avoid repetition [PXS18]. Similarly the Bert model, pretrained in a next sentence prediction setting, encodes knowledge of previously generated tokens through multi-head attention layers in the transformer model [Dev+19; Vas+17]. Both approaches use teacher forcing during training [PXS18; Dev+19].

Nallapati et al. also discovered that the more diverse human summaries from the CNN/Daily Mail datasets are better suited for evaluating abstractive summarization with multiple sentences [Nal+16; Her+15]. Given a benchmark dataset like the CNN/Daily Mail datasets, summarization quality is usually automtically measured using ROUGE [Lin04]. Unigram overlap (ROUGE-1) and bigram overlap (ROUGE-2) with the target summary from the benchmark dataset indicate a generated summary's informativeness and the longest common subsequence (ROUGE-L) can be used to assess fluency [LL19; Lin04].

2.2 Summarization With Pretrained Language Models

Pretrained language models like ELMo, GPT or BERT are motivated by dividing natural language processing tasks into two steps: unsupervised pretraining and task specific, supervised fine-tuning [Pet+18; Rad+18; Dev+19]. With this two step approach, language models can be pretrained unsupervised on large corpora and must only be fine-tuned supervised on a smaller dataset [Dev+19]. For instance the BERT-Base model as used by Liu and Lapata was trained on a corpus containing 3300M words. When adapting BERT for a specific natural language processing task, its parameters are usually jointly fine-tuned with task specific model parameters [Dev+19; LL19].

While pretrained ELMo embeddings have been used to achieve state-of-the-art results in abstractive summarization [EBA19], that approach did not use bidirectional context like from BERT embeddings [Dev+19]. Liu and Lapata complement the pretrained BERT encoder with a transformer decoder consisting of several multi-head attention layers [LL19; Vas+17], following the same encoder-decoder design as See, Liu, and Manning [SLM17]. Their new model is named BERTSUMABS [LL19].

Liu and Lapata overcome BERT's position embedding length limit by introducing more trained position embeddings [LL19]. Though, it is not clear if and in what positions those new embeddings are added to BERT embeddings, or if instead the new embeddings replace BERT embeddings entirely. For simplicity, we assume that new embeddings replace BERT embeddings in the BERTSUMABS model. As the decoder transformer layers have to be learned from scratch, Liu and Lapata propose a fine-tuning schedule with separate learning rates for two Adam optimizers, for the pretrained BERT encoder and the freshly trained decoder: the encoder is trained slower with more linear warmup steps, the decoder is trained faster with less warmup steps [LL19].

For evaluation, Liu and Lapata use BERT-Base as encoder and tokenize text with BERT's WordPiece tokenizer [LL19]. The transformer decoder has 6 layers, each having 768 hidden units and a hidden size for the feed-forward layers of 2048. Dropout is applied before linear layers with probability 0.1. Liu and Lapata also smoothened ground truth labels (i.e., the one-hot probability of each token in the target summary) with a smoothing factor of $\alpha_S = 0.1$ [LL19; Sze+16]. Their model is then trained for 200 000 steps with gradient accumulation every 5 steps [LL19]. During training, Liu and Lapata take snapshots every 2500 steps and afterwards select the best 3 snapshots based on evaluation loss on the validation set [LL19]. Here it is ambiguous, whether the models classification loss or the final ROUGE scores should be chosen for selection. We decided to select based on classification loss for our replication.

For generating the summary using the trained model, Liu and Lapata use beam search with a beam width of 5. They applied a length penalty [Wu+16] with α_L tuned from 0.6 to 1.0 on the validation set and block repeating trigrams [PXS18]. We set α_L to be fixed at 0.8 in order to simplify model evaluation. We believe that the large model size of BERTSUMABS and its complex parameter settings can make fine-tuning difficult [Agh+20] and unaffordable for institutions with a lower

budget on computational resources [Jia+20]. Therefore, in our replication we aim to keep parameters fixed and at their default values.

3 EXPERIMENTAL EVALUATION

We fine-tune and evaluate the BertSumAbs summarization model from the reproduced paper by Liu and Lapata on a A100 GPU using the Flux framework on Julia [Inn+18; Bez+17; LL19]. A scaled-down variant of their TransformerAbs model, that we call TransformerAbsTiny, is trained from scratch on a GeForce MX150 GPU for testing our evaluation workflow. We train both models using the CNN/Daily Mail datasets, but use the preprocessed variant by Liu and Lapata instead of the original data by Hermann et al. [LL19; Her+15]. Our implementation resembles the same encoder-decoder architecture as used by Liu and Lapata, uses the same Bert-Base model for the pretrained encoder, and Bert's WordPiece tokenizer to transform raw text into embeddable tokens [LL19; Dev+19]. Though, we do not tune any hyperparameters of either the model or the optimizers. We also reimplement the beam search algorithm. Due to instabilities in fine-tuning and hardware constraints we were unable to train the model for the full 200 000 steps proposed by Liu and Lapata [LL19]. We assume that the complex hyperparameter settings and large model size used in their paper are prone to fine-tuning issues like we experienced, a problem that has only recently gained more attention [Dod+20; Zha+20; Agh+20].

3.1 Dataset

We train and evaluate the summarization model on the CNN/Daily Mail benchmark dataset using the standard training, test, and validation splits, as crawled by Nallapati et al. [Her+15; Nal+16]. Liu and Lapata preprocess the raw non-anomized CNN/Daily Mail dataset, split sentences with the Stanford CoreNLP framework [Man+14], and release the preprocessed data publicly [LL19]. Our implementation is designed to automatically download those preprocessed CNN/Daily Mail datasets from the Web[1], no manual downloads are needed.

Even though Liu and Lapata claim to truncate input sequences after 512 tokens, we observe sequences longer than 512 tokens in the preprocessed data [LL19]. We use the opportunity to not truncate input sequences, because we find that truncating after only 512 tokens discards a reasonable amount (33 % on average) of context from the news articles [Nal+16].[2] From the preprocessed data we extract pairs of each source article with its target summary, normalize sentence separators, and tokenize with Bert's WordPiece tokenizer [Dev+19].

Like Liu and Lapata, we use the Bert-Base model, pretrained by Devlin et al., as encoder [Dev+19]. That Bert model is also automtically downloaded from the Web along with its WordPiece tokenizer.[3]

[1] https://drive.google.com/open?id=1DN7ClZCCXsk2KegmC6t4ClBwtAf5galI

[2] Arguably, especially the last sentences of a news article can be important to draw the right conclusion in a summary.

[3] Bert is loaded by the Transformers library: https://github.com/chengchingwen/Transformers.jl

3.2 Abstractive Summarizaion Model

Our implementation of the BertSumAbs abstractive summarization model follows the same encoder-decoder architecture as described by Liu and Lapata [LL19]. Input and output tokens are embedded with learned word embeddings and position embeddings with a max length of 4096 tokens. Our transformer decoder is made of 6 transformer layers that each have 768 hidden units with a hidden size of 2048 for feed-forward layers. We apply dropout with a probability of 0.1 before all linear layers, the same parameters as used by Liu and Lapata [LL19]. A linear layer with softmax activation function generates token probabilities for the 30K tokens in Bert's vocabulary [Dev+19]. The full BertSumAbs model contains 182M trainable parameters: 27M for embeddings, 85M for the encoder's transformer layers, 47M for the decoder's transformer layers, and 23M for the generator layer.

We contrast the large BertSumAbs with a smaller, non-pretrained baseline variant, TransformerAbsTiny, that is designed similar to the TransformerAbs model by Liu and Lapata but with even less parameters [LL19]: The randomly initialized model uses 2 transformer layers for the encoder and 2 transformer layers for the decoder. All layers have only 8 hidden units with a hidden size of 4 and dropout probability 0.1. The TransformerAbsTiny is designed to be trainable even on low-performance machines as it only contains 37K trainable parameters.

3.3 Fine-tuning the pretrained model

Following the implementation of [LL19], we planned to fine-tune the BertSumAbs model for 200 000 steps, but with gradient accumulation after each step [LL19]. We save snapshots of the model, its weights, and its optimizers every 2500 steps. Like Liu and Lapata, we implement a custom training schedule with separate learning rates for the pretrained encoder (η_E) and randomly initialized decoder (η_D):

$$\eta_E = 2e^{-3} \cdot \min(\text{step}^{-0.5},\ \text{step} \cdot 20\,000^{-1.5})$$
$$\eta_D = 0.1 \cdot \min(\text{step}^{-0.5},\ \text{step} \cdot 10\,000^{-1.5})$$

Both encoder and decoder are trained using Adam optimizers with the above learning rates, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Cross entropy between the token probabilities predicted by the model and the observed, one-hot ground truth probabilities is used as classification loss for training the models. We do not apply label-smoothing like Liu and Lapata do [LL19].

Unfortunately, on our hardware (using an A100 GPU) we were unable to train the model for the full 200 000 steps but instead had to cancel training after 22 500 steps. Figure 1 shows that the loss when training BertSumAbs quickly begins to oscillate after ony 25 steps. The oscillation continues to the end of training. Zhang et al. suggest that changes in the optimizer can in some cases cause instabilities in the fine-tuning process [Zha+20]. Given that the Bert encoder and transformer decoder of the BertSumAbs model are trained with custom learning rates and Adam optimizers, we ask whether these complex hyperparameter settings are causing the observed fine-tuning issues. A more detailed analysis has to be conducted to test if the issues can be circumvented by training with different

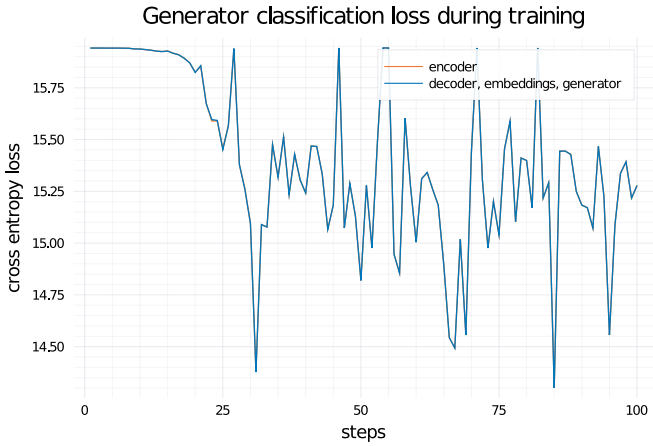

Fig. 1. Cross entropy between predicted token probabilities and ground truth labels for the first 100 training steps of training the BertSumAbs model.

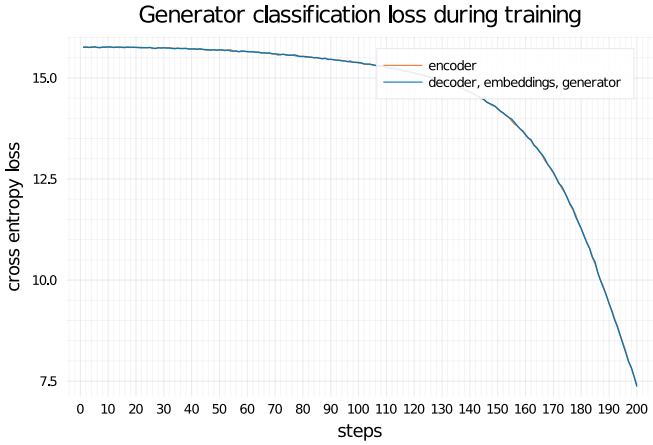

Fig. 2. Cross entropy between predicted token probabilities and ground truth labels for the first 200 training steps of training the TransformerAbsTiny model.

random seeds [Dod+20]. New approaches also fine-tuning pretrained language models for abstractive summarization suggest an improvement by using trust regions [Agh+20]. It is also notable that training the BertSumAbs model on the A100 GPU used about 40GB of memory at a speed of about 9 steps per minute. Though the memory usage is comparable to the memory usage reported by Liu and Lapata [LL19]. The small TransformerAbsTiny model was only trained on the less powerful GeForce MX150 GPU for the first 200 training steps, but as shown in Figure 2, the loss decreases more stable than for the BertSumAbs model.

3.4 Beam Search

In addition to the transformer encoder-decoder model, we also reimplement the beam search algorithm with parameters used by Liu and Lapata. In each

Table 1. Quality labels for assessing summary quality by human annotation.

Label	Description
0	unreadable summary or unrelated to original article
1	readable and related to original article, some redundancy
2	captures exactly the article's main points, no redundancy

iteration of the beam search algorithm we keep track of the best n sequences based on a scoring function s. Here, n denotes the beam width and s is based on the probability $p(x_1, \ldots, x_n)$ of each sequence $\mathbf{x} = [x_1, \ldots, x_n]$. We start with the a sequence only containing the start token and then subsequently expand with new tokens from the vocabulary, until we either see the special end-of-sequence token or hit a configurable maximum length. The probability for a new sequence $\hat{\mathbf{x}} = [x_1, \ldots, x_{n+1}]$ with the new token x_{n+1} is then calculated as the conditional probability

$$p(x_1, \ldots, x_{n+1}) = p(x_{n+1}|x_1, \ldots, x_n)p(x_1, \ldots, x_n)$$

where we get the next-token prediction $p(x_{n+1}|x_1, \ldots, x_n)$ from the model's generator layer. The scoring function s is used to model maximum length $n_{\max}$, length penalization with factor α_L [Wu+16], and to block repeated trigrams [PXS18]:

$$s(x_1, \ldots, x_n) = \begin{cases} 0 & \text{if} n > n_{\max} \\ 0 & \text{if } [x_{n-2}, x_{n-1}, x_n] \text{ occurs} \\ & \quad \text{before in } [x_1, \ldots, x_n] \\ p(x_1, \ldots, x_n) / \frac{(5+n)^{\alpha_L}}{(5+1)^{\alpha_L}} & \text{else} \end{cases}$$

Even though we were unable to test the beam search implementation as we lack a trained model for computing $p(x_{n+1}|x_1, \ldots, x_n)$, we confirm that on a untrained model random nonsense sequences of arbitrary length are generated that all have very low probability.

3.5 Summary Quality

Due to not having a trained model, we are unable to automatically or manually assess summary quality for the BertSumAbs model. We originally planned to automatically measure unigram overlap (ROUGE-1) and bigram overlap (ROUGE-2) between generated and ground truth summaries to assess informativeness as well as the longest common subsequence (ROUGE-L) to assess fluency in terms of the ROUGE framework [Lin04]. Additionally we planned to randomly select a small subset of generated summaries to be judged by human annotators in the three quality levels shown in Table 1. Though, as we are unable to evaluate summary quality without being able to train the summarization model, we cannot argue whether the ROUGE scores reported by Liu and Lapata for the BertSumAbs model are valid or not [LL19].

4 CONCLUSION AND FUTURE WORK

We fail to reproduce the results from Liu and Lapata on abstractive summarization using the BertSumAbs model and identify two main causes: First, the complex model and hyperparameter settings, particularly the use of two separate, customized Adam optimizers for the pretrained Bert encoder and the randomly initialized decoder can lead to instability during training, as has recently been argued similarly in literature [Zha+20]. Second, the model, with 182M trainable parameters, is very large and thus it is difficult to train unless having access to expensive high-performance hardware like Cloud TPUs. We question both the social impact and the environmental implications of using such high computational power for minor improvements on a single task. Even though we cannot confirm the exact causes for the instability in training that we observed, we argue that fine-tuning Bert is not as straight-forward as Liu and Lapata propose. We expect regularizaion to have a positive effect on the fine-tuning, which is probably also easier to implement than the complex two-optimizers setting.

4.1 Future Work

Recently the much larger GPT-3 language model has been introduced. It does require no fine-tuning but instead can be used in a few-shot setting by simple instructions [Bro+20]. We wonder if the GPT-3 model could also be successfully applied to abstractive summarization. On the other hand, smaller models of the Bert architecture emerge that are optimized for more constrainted environments [Tur+19]. If used in a similar architecture as proposed by Liu and Lapata for the BertSumAbs model, we wonder if comparable results on the ROUGE benchmark could also be achieved with Bert-Tiny used as encoder [Tur+19]. As we could not successfully train the BertSumAbs model, we were also unable to tune hyperparameters for the transformer decoder, an evaluation that is missing from the replicated paper [LL19]. With smaller models being more easily trainable, a study of varying model depth and width is an interesting direction for future research.

A APPENDIX

We release our reproduction's source code online under the free MIT license:[4]
https://github.com/heinrichreimer/text-summarization-reproducability
Our source code can most easily be accessed with the included Pluto[5] notebook that guides through training and evaluation interactively.

REFERENCES

[Agh+20] Armen Aghajanyan et al. "Better Fine-Tuning by Reducing Representational Collapse". In: *CoRR* abs/2008.03156 (2020). arXiv: 2008.03156.

[4]The last commit prior to submitting this report for grading had the following SHA-1 hash: `c5e08c5967e366cce3810f2bdf96c24e81810981`
[5]https://github.com/fonsp/Pluto.jl/

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: 3rd International Conference on Learning Representations. ICLR 2015 (San Diego, California, United States, May 7–9, 2015). Ed. by Yoshua Bengio and Yann LeCun. 2015. arXiv: 2008.03156.

[Bez+17] Jeff Bezanson et al. "Julia: A Fresh Approach to Numerical Computing". In: *SIAM Rev.* 59.1 (2017), pp. 65–98. DOI: 10.1137/141000671.

[Bro+20] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems 33*. Annual Conference on Neural Information Processing Systems 2020. NeurIPS 2020 (Virtual, Dec. 6–12, 2020). Ed. by Hugo Larochelle et al. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[Cel+18] Asli Celikyilmaz et al. "Deep Communicating Agents for Abstractive Summarization". In: 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. NAACL-HLT 2018 (New Orleans, Louisiana, United States, June 1–6, 2018). Ed. by Marilyn A. Walker, Heng Ji, and Amanda Stent. Vol. 1. Association for Computational Linguistics, 2018, pp. 1662–1675. DOI: 10.18653/v1/n18-1150.

[Dev+19] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. NAACL-HLT 2019 (Minneapolis, Minnesota, United States, June 2–7, 2019). Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Vol. 1. Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/n19-1423.

[Dod+20] Jesse Dodge et al. "Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping". In: *CoRR* abs/2002.06305 (2020). arXiv: 2002.06305.

[Don+19] Li Dong et al. "Unified Language Model Pre-training for Natural Language Understanding and Generation". In: *Advances in Neural Information Processing Systems 32*. Annual Conference on Neural Information Processing Systems 2019. NeurIPS 2019 (Vancouver, British Columbia, Canada, Dec. 8–14, 2019). Ed. by Hanna M. Wallach et al. 2019, pp. 13042–13054. URL: https://proceedings.neurips.cc/paper/2019/hash/c20bb2d9a50d5ac1f713f8b34d9aac5a-Abstract.html.

[EBA19] Sergey Edunov, Alexei Baevski, and Michael Auli. "Pre-trained language model representations for language generation". In: 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. NAACL-HLT 2019 (Minneapolis, Minnesota, United States, June 2–7, 2019). Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Vol. 1. Association for Computational Linguistics, 2019, pp. 4052–4059. DOI: 10.18653/v1/n19-1409.

[Her+15] Karl Moritz Hermann et al. "Teaching Machines to Read and Comprehend". In: *Advances in Neural Information Processing Systems 28*. Annual Conference on Neural Information Processing Systems 2015. NIPS 2015 (Montreal, Quebec, Canada, Dec. 7–12, 2015). Ed. by Corinna Cortes et al. 2015, pp. 1693–1701. URL: https://proceedings.neurips.cc/paper/2015/hash/afdec7005cc9f14302cd0474fd0f3c96-Abstract.html.

[Inn+18] Michael Innes et al. "Fashionable Modelling with Flux". In: *CoRR* abs/1811.01457 (2018). arXiv: 1811.01457.

[Jia+20] Xiaoqi Jiao et al. "TinyBERT: Distilling BERT for Natural Language Understanding". In: 2020 Conference on Empirical Methods in Natural Language Processing: Findings. EMNLP 2020 (Online Event, Nov. 16–20, 2020). Ed. by Trevor Cohn, Yulan He, and Yang Liu. Association for Computational Linguistics, 2020, pp. 4163–4174. DOI: 10.18653/v1/2020.findings-emnlp.372.

[Kle+17] Guillaume Klein et al. "OpenNMT: Open-Source Toolkit for Neural Machine Translation". In: 55th Annual Meeting of the Association for Computational Linguistics. ACL 2017 (Vancouver, Canada). Ed. by Mohit Bansal and Heng Ji. Association for Computational Linguistics, 2017, pp. 67–72. DOI: 10.18653/v1/P17-4012.

[Lin04] Chin-Yew Lin. "ROUGE: a Package for Automatic Evaluation of Summaries". In: Text Summarization Branches Out, Post-Conference Workshop of ACL 2004 (Barcelona, Spain, July 2004). Association for Computational Linguistics, 2004, pp. 74–81. URL: https://aclweb.org/anthology/W04-1013.

[LL19] Yang Liu and Mirella Lapata. "Text Summarization with Pretrained Encoders". In: 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing. EMNLP-IJCNLP 2019 (Hong Kong, China, Nov. 3–7, 2019). Ed. by Kentaro Inui et al. Association for Computational Linguistics, 2019, pp. 3728–3738. DOI: 10.18653/v1/D19-1387.

[Man+14] Christopher D. Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit". In: 52nd Annual Meeting of the Association for Computational Linguistics. ACL 2014 (Baltimore, Maryland, United States, June 22–27, 2014). Association for Computer Linguistics, 2014, pp. 55–60. DOI: 10.3115/v1/p14-5010.

[Nal+16] Ramesh Nallapati et al. "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond". In: 20th SIGNLL Conference on Computational Natural Language Learning. CoNLL 2016 (Berlin, Germany, Aug. 11–12, 2016). Ed. by Yoav Goldberg and Stefan Riezler. ACL, 2016, pp. 280–290. DOI: 10.18653/v1/k16-1028.

[Pet+18] Matthew E. Peters et al. "Deep Contextualized Word Representations". In: 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. NAACL-HLT 2018 (New Orleans, Louisiana, United States, June 1–6,

2018). Ed. by Marilyn A. Walker, Heng Ji, and Amanda Stent. Association for Computational Linguistics, 2018, pp. 2227–2237. DOI: 10.18653/v1/n18-1202.

[PXS18] Romain Paulus, Caiming Xiong, and Richard Socher. "A Deep Reinforced Model for Abstractive Summarization". In: 6th International Conference on Learning Representations. ICLR 2018 (Vancouver, British Columbia, Canada, Apr. 30–May 3, 2018). OpenReview, 2018. URL: https://openreview.net/forum?id=HkAClQgA-.

[Rad+18] Alec Radford et al. "Improving Language Understanding by Generative Pre-Training". In: (2018).

[SAR20] Alexandra Savelieva, Bryan Au-Yeung, and Vasanth Ramani. "Abstractive Summarization of Spoken and Written Instructions with BERT". In: KDD 2020 Workshop on Conversational Systems Towards Mainstream Adoption co-located with the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. SIGKDD 2020 (Virtual Workshop, Aug. 24, 2020). Ed. by Giuseppe Di Fabbrizio et al. Vol. 2666. CEUR Workshop Proceedings. CEUR-WS.org, 2020. URL: http://ceur-ws.org/Vol-2666/KDD%5C_Converse20%5C_paper%5C_11.pdf.

[SLM17] Abigail See, Peter J. Liu, and Christopher D. Manning. "Get To The Point: Summarization with Pointer-Generator Networks". In: 55th Annual Meeting of the Association for Computational Linguistics. ACL 2017 (Vancouver, British Columbia, Canada, July 30–Aug. 4, 2017). Ed. by Regina Barzilay and Min-Yen Kan. Vol. 1. Association for Computational Linguistics, 2017, pp. 1073–1083. DOI: 10.18653/v1/P17-1099.

[Sze+16] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2016 (Las Vegas, Nevada, United States, June 27–30, 2016). IEEE Computer Society, 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308.

[Tor14] Juan-Manuel Torres-Moreno. *Automatic Text Summarization*. Wiley, 2014. ISBN: 978-1-848-21668-6.

[Tur+19] Iulia Turc et al. "Well-Read Students Learn Better: The Impact of Student Initialization on Knowledge Distillation". In: *CoRR* abs/1908.08962 (2019). arXiv: 1908.08962.

[Vas+17] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30*. Annual Conference on Neural Information Processing Systems 2017. NIPS 2017 (Long Beach, California, United States, Dec. 4–9, 2017). Ed. by Isabelle Guyon et al. 2017, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[Wu+16] Yonghui Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144 (2016). arXiv: 1609.08144.

[Zha+20] Tianyi Zhang et al. "Revisiting Few-sample BERT Fine-tuning". In: *CoRR* abs/2006.05987 (2020). arXiv: 2006.05987.