Team Matus and Francesco @ AutoMin 2021: Towards neural summarization of meetings

Matúš Žilinec, Francesco Ignazio Re

ETH Zürich, Switzerland

matus.zilinec@inf.ethz.ch, franre@student.ethz.ch

Abstract

As online meetings are becoming increasingly ubiquitous, there is an increasing demand to record the main outcomes of these meetings for future reference. Automatic summarization of meetings is a challenging, yet relatively unexplored natural language processing task with a wide range of potential applications. This paper describes our submission to the First Shared Task on Automatic Minuting at Interspeech 2021. In contrast to previous research focused on the summarization of narrated documents, we examine the specifics of bullet-point spoken language summarization on the AutoMin dataset of online meetings in English. Furthermore, we investigate whether existing abstractive summarization systems can be transferred to this new domain. In this regard, we develop a minuting pipeline based on the state-of-the-art PEGASUS summarization model. This includes pre-processing of conversational data, few-shot transfer learning using reference minutes generated by human annotators, filtering and post-processing of the resulting candidate summaries into a suitable bullet-point minutes format. We conclude by evaluating the completeness and shortening aspects of our system, and discuss its limitations and potential future research directions.

Index terms: automatic minuting, summarization, transfer learning

1. Introduction

Recent advances in Natural language processing have provided the means to automate tedious tasks previously done exclusively by humans. Given the ever increasing amount of textual data available on-line, efforts have been made to develop models for automatic text summarization. Most state-of-the-art models are trained for summarization of well formatted text, such as news articles or blog posts. This work focuses instead on the area of automatic minuting, which comprises of generating bullet point summaries from transcripts of conversations, such as online team meetings, podcasts, or parliamentary sessions. This task is closely related to summarization with the added difficulty of processing spoken language. The main challenges that arise are the following:

The resulting summary should contain information from multiple speakers that interact through free-form dialogue, which makes the summary dependent on the context of the conversation. This is in contrast to e.g. news summarization with a single narrator, where facts are spoken in simple declarative sentences. Furthermore, transcribed meetings are not redacted and their format may vary according to context. Different speakers use different language styles and vocabulary, often making mistakes or false statements, which may be corrected by another speaker.

We explore this area by taking part in the First Shared Task on Automatic Minuting at Interspeech 2021 [1], which provides a new corpus of meetings in English.

2. Related work

Summarization is a task in NLP that concerns itself with the shortening of a document while preserving the most important information. In recent years it has gained attention with the release of several datasets such as CNN/DailyMail [2], arXiv, PubMed [3], XSum [4] and others. Current summarization systems can be classified into two groups, extractive and abstractive [5]. Extractive models perform summarization by selecting a subset of the input document, whereas the latter make use of a generative model to generate arbitrary paraphrases of the document's content. [6] have shown that both of these approaches are useful in different contexts. Most prior work on abstract summarization is based on the sequence-to-sequence framework with various implementations of the encoder and decoder, such as recurrent neural networks [7], [8], or Transformer [9]. For longer inputs, reinforcement learning has been found to produce more readable outputs [10]. Applications of self-supervised pre-training of Transfomers have shown significant performance improvements, with the added flexibility of allowing few-shot transfer to different tasks [11], [12]. Among the most recent approaches to summarization, [13] introduced a transformer based architecture (T5) that uses a text-to-text approach, and [14] presented a new sequence-to-sequence pre-training model, ProphetNet. Finally, [15] achieved state-of-the-art performance by using a custom pre-training objective, where the model is trained to recover important sentences masked from a document.

3. Dataset

We participate in the main track of the AutoMin shared task, the objective of which is to automatically create minutes from meeting transcripts. The task runs in two languages, namely English and Czech, and a separate meeting corpus is provided for each language. Since we are mostly interested in the transfer of existing neural mod-

els to this domain, we limit our submission to the English language, for which these models are readily available.

3.1. Dataset Description

The AutoMin training dataset consists of transcripts generated from the audio recordings of a research group's online meetings in English language. In total, there are 85, 10 and 28 dialogues provided as training, validation and testing data respectively. Additionally, the dataset contains one or more reference minutes written by a human annotator for each transcript in the training dataset. Each of the transcribed meetings is represented by a single plain text file, where each line corresponds to the corrected speech recognition output of a single utterance in the conversation. In some cases, utterances are prefixed with the name of the speaker. Notably, as the dataset was collected with real-world applications in mind, the quality of the raw text is rather low. We will now summarize the most significant quality issues that we have observed in the data.

3.2. Dataset Analysis

First, as is expected in the dialogue domain, the text contains frequent filler words, unfinished utterances, repeated parts of speech, and corrections of previous sentences. Second, multiple topics are discussed in each meeting. Some of these may not be relevant for the summary, such as informal introductions and smalltalk present throughout the discussion. Third, the topics discussed in the meetings are often of a technical nature, and summarizing them may require prior knowledge of the topic and other context, such as the current state of the project at the time of the meeting. This is especially an issue in online presentations, where the speaker is sharing their screen with the other participants.

Yet another issue is the abundance of named entities in the dataset. All named entities, including names of persons, companies and events, are replaced by IDs, which are not easily handled by word embedding methods. Furthermore, the meetings as well as the individual utterances of each speaker vary widely in their length. Finally, meeting participants include non-native English speakers, whose speech is automatically transcribed. This results in the text containing frequent grammatical mistakes and sentences with illegal structure.

Reference minutes in the AutoMin dataset were manually created by human annotators, and vary in their length and format. We observe two distinct minuting styles. One group of annotators prefer to use short, bullet point lists of typically 5 to 20 main takeaways from the discussion. The remaining annotators prefer describing discussed topics using full sentences. The resulting minutes are typically longer than 20 sentences.

4. Summarization model

We have previously described several distinct approaches to summarization, including abstractive and extractive. We believe that extractive summarization is unsuitable for our task, mainly because of the need to convert dialogue into narrative text. Among abstractive models, we decided to opt for the PEGASUS model (Pre-training with Extracted Gap-sentences for Abstractive Summarization) [15], which achieves state-of-the-art performance on current summarization benchmarks. The model implements a transformer encoder-decoder architecture and was pre-trained using a self-supervised objective on a large corpus of web-scraped documents. We decided to use this model based on its novel pre-training objective, which relies on estimating the importance of each sentence in the training text. Sentences deemed as important are masked from the input and the model is tasked with recovering them.

Furthermore, its implementation and pre-trained weights are available online and backed by a wide community of users, making it suitable for future real-world applications. We adapt this model to our domain via fine-tuning, pre- and post-processing.

5. Preprocessing

Because of the low quality of the text present in the dataset, we employ several preprocessing steps before feeding the data into a neural network. First, we apply regular expressions separately to each utterance. We remove common filler word variants, such as *"uhm"*, duplicated tokens, such as *"actually, actually"*, unfinished words ending with the hyphen "-", such as *"well"*, and certain HTML tags denoting noises, such as *"<laugh>"*. Furthermore, we parse and remove the speaker ID prefixing the utterance. In case no speaker ID is provided, the sentence is assumed to have been uttered by the previous speaker. We uniform dates both in dialogues and in the manual annotations. Dates are reformatted as *DD/MM/YY*.

5.1. Narrative format with co-reference resolution

Coreference resolution is applied as an additional preprocessing step, with the aim of adapting the text to a narrative form. Specifically, we replace pronouns such as *"that", "you", "I"* with the corresponding named entities to aid the model in understanding associations between sentences. This preprocessing step was only used in our secondary submission, described in Results.

We use a simple but robust rule-based algorithm, replacing all first-person pronouns and adjectives with the name of the speaker, when provided. We also change "to be" and "to have" conjugations in case the verb immediately precedes or succeeds the subject. As an example, the following line

(PERSON6) : - so I managed to share the screen and, uh, please take a look at my screen now

would become

So (PERSON6) managed to share the screen and, uh, please take a look at (PERSON6) screen now.

5.2. Dialogue partitioning

Followed by this, we create inputs to the summarization model. Because of memory constraints, each document is fed into the model in a number of separate input chunks. The output text for each input chunk is then treated as a single candidate minute. Input chunks are generated by sequentially iterating over utterances and merging or splitting them according to these criteria:
a) Sequential utterances from a single speaker are concatenated into a single chunk. This ensures that the model processes the entire thought expressed by the person in a single input.
b) Input chunks shorter than 4 tokens are removed, since these are most often short responses not useful for summarization.
c) We aim to produce inputs of approximately bounded length to ensure a reasonable content-to-length ratio of outputted summary for all parts of the input transcript. Thus, we set a minimal and maximal permissible number of input tokens, and merge or split adjacent input chunks to meet these constraints.

In order to make training and evaluation on this dataset possible, it is also necessary to align each input chunk to a corresponding bullet-point in the reference minutes. We perform this alignment separately for each document. We begin by removing stopwords using NLTK and applying TF-IDF to reference minutes. This results in a list of the most significant (1,2)-grams present in each reference minute. All input chunks and reference minutes are then converted into a bag-of-words representation and the overlap of (1,2)-grams is calculated for each pair. Each input chunk is subsequently aligned to exactly one reference with the highest overlap.

6. Evaluation

6.1. Shared task metrics

In the AutoMin shared task, submissions are judged on the testing dataset with hidden labels primarily via human evaluation, supplemented by automatic metrics. The automatic evaluation relies on calculating ROUGE-L [16] between model outputs and reference summaries. The manual evaluation was done by two annotators, who ranked the adequacy, fluency and grammatical correctness of each submission on a scale from 1 to 5 for each document.

6.2. Proposed metrics

In addition to relying on shared task evaluation, we used our own metrics for model selection on the dev dataset. However, the nature of the task makes automatic evaluation challenging. First, human-written minutes vary greatly between different human annotators. Since the type of reference minutes is conditional on the annotator, its length will necessarily differ from the length of the output of our model. Second, there is no 1-to-1 correspondence between the generated minutes and reference text. For this reason, it is necessary to align output sentences and references in evaluation.

Ideally, we would like to measure the number of "facts" in reference minutes that are also present in the output of the model. At the same time, we would like to penalize the output length to ensure the model produces the shortest possible summary. We begin by treating every line of the reference minutes as one such "fact", and aligning these facts to the the output of the model per-line using the alignment method described in Section 5.2. We then propose the following way of calculating precision and recall.

Let $\mathcal{A}$ be the set of sentences outputted by a summarization model, and $\mathcal{B}$ the set of reference sentences for a single document. Then, let $x \in A$ and $y \in B$. We define $\alpha \subset \mathcal{A} \times \mathcal{B}$ as the set of all pairs (x, y) such that x was aligned to y. Importantly, with our alignment method, every output sentence is aligned to exactly one reference sentence, so that $\forall(x, y, y') : (x, y) \in \alpha \wedge (x, y') \in \alpha \Rightarrow y = y'$. We define precision and recall for the minuting task respectively as

$$P = \frac{|\{\forall y : \exists x, (x, y) \in \alpha\}|}{|\mathcal{A}|},$$

$$R = \frac{|\{\forall y : \exists x, (x, y) \in \alpha\}|}{|\mathcal{B}|}.$$

Therefore, we assume that for a *complete* model summarizing all important facts from the document, each reference sentence would be aligned to at least one output sentence, and recall would be maximized. Conversely, for a trivial model not depending on the input, only one reference sentence would be aligned, minimizing recall.

The precision score then acts as a brevity penalty, such that for a *shortening* model outputting only relevant sentences, each reference sentence would align to only one output sentence, maximizing the precision. In contrast, precision would be minimal for a model outputting the whole input document, where each reference sentence would be aligned to a number of output sentences.

Thus, the resulting F1 score measures the *completeness* and *shortening* aspects of a model. Aside from this, we also report the standard BLEU [17] and ROUGE-L scores.

7. Experiments

7.1. Setup

In all our experiments, we used Python 3.6, NLTK 3.6.2, PyTorch 1.4.0 and Huggingface transformers 3.5.1. Fine-tuning was performed on NVIDIA GPUs with 15GB VRAM. The pretrained model that we use can be obtained from the Huggingface repository[1]. Our code can be downloaded from:

https://github.com/mzilinec/automatic-minuting.

[1] https://huggingface.co/google/pegasus-wikihow

	Precision	Recall	F1	BLEU	ROUGE-L
Baseline- Copy Input	1.48	70.19	2.09	0.65	3.33
Baseline - PEGASUS	12.18	37.89	18.43	0.84	10.53
Fine-tune base	15.57	48.45	23.56	1.28	9.36
Fine-tune + decoder-fix	14.57	45.34	22.05	1.46	12.07
Fine-tune + decoder-fix + post	**21.32**	**36.02**	**26.79**	**2.06**	**18.84**

Table 1: *Evaluation results on the dev dataset.*

7.2. Baselines

Before attempting to fine-tune PEGASUS on the minuting domain, we establish two simple baselines. As the trivial baseline, we treat the document itself as its summary. This baseline results in a high recall and low precision, due to the amount of unnecessary information preserved in the summary.

As the second baseline, we use a pre-trained PEGASUS Base model without any modifications "out of the box". While this model receives a higher score using automated metrics, after a manual inspection, the outputs are clearly nonsensical. The model simply copies certain tokens from the input and invents an entirely new narrative containing these tokens, similar in its topic to the pre-training data. Only a small subset of generated sentences contain valid summaries of the input document.

7.3. Transfer learning

We continue by carrying out fine-tuning of the PEGASUS Base model on the preprocessed AutoMin dataset. The model is trained to output a single reference minute given a single input chunk in a standard sequence-to-sequence fashion. Each training example consists of an input chunk aligned to the most similar output reference minute. We find that training using the full training dataset doesn't produce good results. This is caused by the fact that the number of input sentences is much higher than the number of reference minutes, which means most input chunks do not correspond to any output sentence. To avoid training on unrelated input-output sentence pairs, we only keep those sentence pairs for which alignment score is ≥ 0.6. This step, although necessary, shrinks the number of training examples to only ~ 1400. Training is performed for one epoch with $\text{batch size} = 1$ due to memory constraints. We use the standard PyTorch SGD optimizer with $lr = 2 * 10^{-5}$ and $\text{momentum} = 0.9$. We observe that training tends to be unstable and may produce an unusable model. For this reason, we repeat training 5 times with identical parameters and different order of training examples. We then manually choose the best run as our submission, listed as *Fine-tune base* in 8.1.

7.4. Decoder optimization

While the fine-tuned PEGASUS model produces valid summaries, we do not find the output satisfactory. Most importantly, the text is generated in the first person and contains personal pronouns. Additionally, the model tends to repeat n-grams at the end of each sentence instead of generating the end token. We experiment with optimizations of the decoder to alleviate these issues. Specifically, we use a list of selected personal pronouns and filler words and prevent the decoder from generating any of these tokens. Additionally, we prevent the decoder from generating any 3-gram more than once. This results in an increase of the quality of generated text (listed as *Fine-tune + decoder-fix* in results). Lastly, we set these decoder parameters:
$\text{num\_beams} = 3$, $\text{early\_stopping} = true$, $\text{repetition\_penalty} = 1.2$, $\text{length\_penalty} = 0.7$, $\text{early\_stopping} = true$.

7.5. Postprocessing

As explained in section 5.2, the model outputs a set of one-sentence summaries for each dialogue, where each sentence corresponds to one of the input chunks. As most input chunks don't contain interesting content, this implies that most output sentences will not be relevant for the summary. For this reason, we employ post-processing to filter out minutes that do not contain relevant information. We again use a technique based on sentence similarity. Specifically, we create a TF-IDF representation of each document and select n words with the highest scores, using a manually selected threshold $k = 0.5$. After this ranking, we keep only the generated minutes that contain at least α of this set of words. For the final submission we set $\alpha = 2$. The choice of α, k controls the number and subset of output minutes kept, and can be used to adjust the tradeoff between precision and recall. In case a generated output minute consists of multiple statements (delimited by *<SEP>* tokens), only the first statements is kept, as we observed that the remaining sentences are usually irrelevant. The resulting minutes are listed as *Fine-tune + decoder-fix + post* in 8.1).

8. Results

8.1. Our evaluation

In this section, we will provide results of our evaluation using standardized metrics as well as manual analysis of the generated text. Evaluation results are summarized in Table 1. We use the F1 metric to rank the overlap between reference and output in terms of factual statements, as described in Section 6. As complementary metrics, we use BLEU and ROUGE-L, which measure the overlap of individual n-grams. Therefore, we use these metrics to evaluate the overall syntactic similarity. For both BLEU and ROUGE-L, each document was treated as a single line. BLEU scores were calculated using Sacrebleu [18].

Surprisingly, baseline PEGASUS achieves competitive scores (especially ROUGE), even though it produces mostly unrelated outputs. This is an argument towards using semantics-aware metrics in the future. We have experimented with using such a metric, namely BLEURT [19]. However, it didn't correlate well with sentence similarity on our dataset and thus we don't include it in evaluation. It is visible that fine-tuning results in a significant recall increase over baseline PEGASUS, as well as a higher average token overlap with the reference than both baselines, measured by BLEU. Decoder optimizations result in a lower F1 score, and increased BLEU. This matches our expectation, since optimizing the decoder makes the outputs syntactically closer to the reference minutes, but information is lost, as the decoder is restricted from generating maximum-likelihood sentences. Lastly, postprocessing results in lower recall, due to the lower number of final sentences kept. Nevertheless, precision and overall F1 score increase, as well as BLEU and ROUGE. This supports our claim that postprocessing can be used to filter the most relevant summarizing sentences from the output of a transformer-based model. To illustrate this length difference, we provide a comparison of average output lengths in Table 2.

Summary-to-reference ratio	# Sentences
Baseline - Copy Input	4980%
PEGASUS	321%
PEGASUS + post-processing	175%

Table 2: *Ratios between the number of output and reference sentences. After filtering, our system still produces outputs almost double the length of reference minutes.*

8.2. Shared task submissions

In total, we sent 3 submissions to the Automin shared task. The first submission is *Fine-tune Base*, as described in last section. The second submission, *Fine-tune Coref*, differs in using PEGASUS-Large instead of PEGASUS-Base, as well as co-reference resolution, described in 5.1. Our final submission, *Fine-tune Final*, corresponds to *Fine-tune + decoder-fix + pos* described in 8.1. *Fine-tune Coref* and *Fine-tune Final* are **late** submissions. Results of automatic evaluation on the test set are shown in Table 3. *Fine-tune Coref* (PEGASUS-Large) achieves a higher Rouge score than PEGASUS-Base models. The effect of improvements included in *Fine-tune Final* is not reflected in Rouge. Human evaluation results are shown in Table 4. Notably, our final (late) submission was ranked more positively than the base submission by the annotators.

8.3. Examples of generated text

We now provide a typical example of a summary generated by the baseline as well as fine-tuned models. It is visible that while the fine-tuned models have generated a more cohherent summary in this example, the generated summary is still factually inaccurate.

Submission	Rouge-1	Rouge-2	Rouge-L
Fine-tune Base	17.0	3.4	9.1
Fine-tune Coref$^{\mathbf{L}}$	21.5	4.1	11.8
Fine-tune Final$^{\mathbf{L}}$	17.5	3.7	9.7
Avg. score	16.7	3.5	9.5
Best score	28.2	6.5	15.9

Table 3: *Automin automatic evaluation results. Late submissions are marked by* ***L****. Average and best scores across all submissions are provided for comparison.*

Submission	Adequacy	Gramm. Corr.	Fluency
Fine-tune Base	2.55	2.91	2.27
Fine-tune Coref$^{\mathbf{L}}$	2.68	3.18	2.73
Fine-tune Final$^{\mathbf{L}}$	2.82	3.50	3.09
Avg. score	2.37	3.10	2.62
Best score	4.25	4.45	4.27

Table 4: *Automin human evaluation results, with scores averaged across documents and annotators. Late submissions are marked by* ***L****.*

Input chunk
Can you give us a date? Mid June seems kind of late. Okay, yes, exactly. Yeah, let's do it earlier. Yep, yeah, yeah. So the 8th, the 8th of June. So that should be the end of the review, kind of, right? Or no. It sho- No, sorry, sorry. Yeah, yeah. The beginning, the beginning. This should be delay. Yeah, yeah. It is the first draft. First draft, yes, but complete draft. Yeah, So then you basically have a week (reviewer) two week to fix it. And a week for no further every week spare - for final tracks from the coordinator.
Output (Pegasus baseline)
So we have the first draft, the beginning, the beginning, the middle, the middle, the beginning, the end, the coordinator, and now you.
Output (Fine-tune base)
Request a date for the first draft, the beginning, the beginning, and the end of the review (reviewers) for final tracks from the coordinator.
Output (Fine-tune + decoder-fix)
Ask for a date for the first draft (reviewer) and final tracks from the coordinator (coordinator).<n>Give us a week (Reviewer) two week to fix it.

9. Conclusion

In this work, we have presented our submission to the First Shared Task on Automatic Minuting. We conclude that neural models trained on the task of abstractive summarization might be viable for automatic minuting of meetings. Nevertheless, there are certain challenges that need to be addressed in the future. A major issue for current models is the sparsity of information content and the length of transcribed meetings. Robustness of minuting systems with respect to low quality input data is yet another issue. We believe that it would be possible to account for these inherent characteristics of the minuting domain by adapting current neural architectures, e.g. by using Transformer variants with attention suitable for long texts, and by pretraining on larger speech corpora. Finally, there is a need for a more concrete definition of the minuting task in terms of evaluation and ground truth labels.

10. References

[1] T. Ghosal, O. Bojar, M. Singh, and A. Nedoluzhko, "Overview of the first shared task on automatic minuting (automin) at interspeech 2021," in *Proceedings of the First Shared Task on Automatic Minuting at Interspeech 2021*, 2021, pp. 1–25. [Online]. Available: http://dx.doi.org/10.21437/AutoMin.2021-1

[2] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *NIPS*, 2015.

[3] A. Cohan, F. Dernoncourt, D. S. Kim, T. Bui, S. Kim, W. Chang, and N. Goharian, "A discourse-aware attention model for abstractive summarization of long documents," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 615–621. [Online]. Available: https://www.aclweb.org/anthology/N18-2097

[4] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 1797–1807. [Online]. Available: https://www.aclweb.org/anthology/D18-1206

[5] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 379–389. [Online]. Available: https://www.aclweb.org/anthology/D15-1044

[6] G. Carenini, R. Ng, and A. Pauls, "Multi-document summarization of evaluative text," in *11th Conference of the European Chapter of the Association for Computational Linguistics*. Trento, Italy: Association for Computational Linguistics, Apr. 2006. [Online]. Available: https://www.aclweb.org/anthology/E06-1039

[7] P. Li, W. Lam, L. Bing, and Z. Wang, "Deep recurrent generative decoder for abstractive text summarization," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 2091–2100. [Online]. Available: https://www.aclweb.org/anthology/D17-1222

[8] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 1073–1083. [Online]. Available: https://www.aclweb.org/anthology/P17-1099

[9] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, "Unified language model pre-training for natural language understanding and generation," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://papers.nips.cc/paper/9464-unified-language-model-pre-training-for-natural-language-understanding-and-generation.pdf

[10] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=HkAClQgA-

[11] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[12] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[13] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2020.

[14] Y. Yan, W. Qi, Y. Gong, D. Liu, N. Duan, J. Chen, R. Zhang, and M. Zhou, "Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training," *CoRR*, vol. abs/2001.04063, 2020. [Online]. Available: https://arxiv.org/abs/2001.04063

[15] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," 2020.

[16] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81. [Online]. Available: https://www.aclweb.org/anthology/W04-1013

[17] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02. USA: Association for Computational Linguistics, 2002, p. 311–318. [Online]. Available: https://doi.org/10.3115/1073083.1073135

[18] M. Post, "A call for clarity in reporting BLEU scores," in *Proceedings of the Third Conference on Machine Translation: Research Papers*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 186–191. [Online]. Available: https://www.aclweb.org/anthology/W18-6319

[19] T. Sellam, D. Das, and A. Parikh, "BLEURT: Learning robust metrics for text generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 7881–7892. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-main.704