Does Pretraining for Summarization Require Knowledge Transfer?

Kundan Krishna, Jeffrey Bigham, Zachary C. Lipton
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
{kundank,jbigham,zlipton}@andrew.cmu.edu

Abstract

Pretraining techniques leveraging enormous datasets have driven recent advances in text summarization. While folk explanations suggest that *knowledge transfer* accounts for pretraining's benefits, little is known about why it works or what makes a pretraining task or dataset suitable. In this paper, we challenge the *knowledge transfer* story, showing that pretraining on documents consisting of character n-grams selected at random, we can nearly match the performance of models pretrained on real corpora. This work holds the promise of eliminating upstream corpora, which may alleviate some concerns over offensive language, bias, and copyright issues. To see whether the small residual benefit of using real data could be accounted for by the structure of the pretraining task, we design several tasks motivated by a qualitative study of summarization corpora. However, these tasks confer no appreciable benefit, leaving open the possibility of a small role for knowledge transfer.[1]

1 Introduction

Despite the widespread success of pretrained models when fine-tuned on diverse downstream NLP tasks, such as summarization (Qi et al., 2020; Raffel et al., 2020), question answering, sentiment analysis etc (Yang et al., 2019), scientific explanations for these benefits remain unknown. Several works have claimed that pretrained models learn linguistic knowledge from the pretraining corpus (Lina et al., 2019; Tenney et al., 2019; Manning et al., 2020), leading to a popular, but unproven hypothesis that credits knowledge transfer for the improvements seen on downstream tasks. However, several recent findings test the plausibility of this account. For example, benefits of pretraining have been observed even when the upstream text has no syntactic structure (Sinha et al., 2021) and others have shown benefits even when the upstream corpus is from a different domain entirely, such as music (Papadimitriou and Jurafsky, 2020) or amino acid sequences (Chiang and Lee, 2020)

In this work, we show that, surprisingly, pretraining objectives previously demonstrated to be helpful for summarization (Zou et al., 2020), continue to deliver significant benefits even when applied on text consisting of randomly sampled nonsense words. Because the text consists of nonsense words sampled independently and uniformly, it seems difficult to fathom a credible argument that the synthetic corpus encodes linguistic knowledge in any relevant sense. Nevertheless, when pretraining transformer-based sequence-to-sequence models using this nonsense text, we achieve significant performance boosts on multiple downstream summarization benchmarks that nearly match the performance of pretrained transformers.

Remarkably, when pretraining with synthetic tasks, using real data offers no benefit over the nonsense data, on multiple summarization benchmarks. Thus, we investigate whether a pretraining task better aligned with the demands of summarization might close this residual gap. We design a collection of pretraining tasks inspired by some of the basic primitive operations that appear to be common routines required in order to create real-world summaries. We carried out an extensive survey of public summarization datasets spanning different domains, and catalogued several elementary operations that were frequently invoked in producing summaries (e.g., extract sentences on a specific topic, or determine the most frequent among a set of relevant terms). In our proposed pretraining corpus, the summary is created by carrying out these elementary operations on the input. However, we find that our pretraining tasks deliver comparable performance gains to those proposed in Zou et al. (2020) leaving the small gap open. On

[1]The code and the datasets used in the paper are available at https://github.com/acmi-lab/pretraining-with-nonsense

Findings of the Association for Computational Linguistics: EMNLP 2021, pages 3178–3189
November 7–11, 2021. ©2021 Association for Computational Linguistics

CNN-Dailymail and Rotowire benchmarks, where median summary lengths are 73 and 456 tokens respectively, using our pretraining tasks with nonsense text results in achieving on average 95% of the performance gain in ROUGE-1 that standard T5 pretrained models enjoy relative to randomly initialized T5. By contrast, on XSum and Rottentomatoes, where summaries are shorter (29 and 32 tokens respectively), we realize a relatively modest 37% of the benefit on average.

The takeaways from our results are two-fold: First, these results challenge our understanding of why pretraining helps in summarization, suggesting that a large portion of the benefits seen may not be due to any knowledge transfer, but simply better initialization from an optimization perspective. Second, the ability to realize the benefits of pretraining without using real-world data could alleviate concerns regarding bias, offensive speech, and intellectual property associated with using web-scale pretraining corpora of unknown provenance (Davidson et al., 2019; Bordia and Bowman, 2019).

2 Related Work

Recently, multiple pretrained models have shown remarkable performance on text summarization. These models have been pretrained on real data with diverse denoising tasks, including masked language modeling (Raffel et al., 2020), text infilling (Zhang et al., 2020), and sentence reordering (Lewis et al., 2020), among others. While these pretraining objectives have shown benefits across multiple NLP tasks, Zou et al. (2020) proposed a set of three denoising pretraining tasks that are specifically motivated by summarization and deliver performance comparable to previous pretrained models. Our paper shows that the pretraining tasks in Zou et al. (2020) improve summarization performance even if the pretraining corpus is artificial and does not encode any linguistic structure.

Our work extends a growing body of scientific literature that questions commonly-held beliefs about what properties of a pretraining corpus lead to improvements on different downstream tasks. Recently, Sinha et al. (2021) showed that word order in pretraining documents has negligible impact on downstream performance on the GLUE benchmark. Even pretraining on sequences from different modalities such as Java code and amino acid sequences (Chiang and Lee, 2020) have shown benefits on GLUE benchmark, Similarly, for the task of language modeling, pretraining on musical scores, or even artificial sequences of nested parentheses has shown to achieve better perplexity on a human language (Papadimitriou and Jurafsky, 2020). Our results go further—here the source documents contain no natural data at all, nor do they exhibit any non-trivial structure.

Recently, some machine learning theory literature has begun to question the mechanism by which transfer learning works. For example, Neyshabur et al. (2020) attribute the benefits to low-level statistics of the data and optimization considerations rather than feature reuse. In other related work, Maennel et al. (2020) show that networks pretrained on randomly labeled data sometimes enjoy considerable performance improvements on downstream tasks.

3 Generating the Nonsense Corpus

For generating the nonsense pretraining corpus, we use an artificial vocabulary to create base documents that has little resemblance to any real language. Our vocabulary simply consists of the first 5000 3-letter character combinations using the English alphabet in lexical order starting from the right (*aaa, baa, caa, ..., aab, bab, ...*). Each sentence is generated by sampling each word in it independently from the uniform distribution over the entire vocabulary, and ending it with a period (see Figure 1 for a sample nonsense document). The length of each sentence is selected uniformly from 5 to 15 words. The number of sentences per document is selected according to the pretraining task that it is used for. For the pretraining tasks proposed in Zou et al. (2020), we sample sentences until the document reaches 512 tokens in length. For our pretraining tasks (introduced later), number of sentences in a document is decided by sampling uniformly from 7 to 13 sentences.

4 STEP Pretraining Tasks

STEP pretraining tasks are a collection of 3 tasks defined by Zou et al. (2020). Next Sentence Generation (NSG) provides the first half of a document as input and the target is to generate the latter half. Sentence Reordering (SR) presents a document with its sentences shuffled in random order, and requires generating the original document with correct sentence order. Masked Document Generation (MDG) masks out a contiguous sequence of tokens in the base document and requires generating the original document while correctly filling-in the masked tokens. More details and hyperparameters can be found in the original paper.

5 Our Pretraining Tasks

To develop our pretraining tasks, we first undertook a qualitative analysis of existing summarization datasets. We surveyed all summarization papers published in the last 10 years with more than 25 citations, cataloguing a list of the summarization datasets that were used in them. We observed that datasets can be grouped together according to domain (e.g., news and conversations). We grouped the 28 retrieved datasets into 14 domains (see the Appendix, Table 9). We selected a single dataset from each domain to analyze what summaries consist of and what *skills* their creation requires.

From each selected dataset, we manually inspected ten randomly sampled input-summary pairs, looking for primitive subtasks that seem to express *skills* (informally) that are required in order to create the summaries demanded by this dataset for at least two of the ten instances. Since we need to create artificial input-summary pairs for each subtask, we only chose subtasks for which it was possible to create large number of such artificial pairs. For example, in the Samsum dataset (Gliwa et al., 2019) which requires summarizing conversations between people, a frequently necessary subtask is to infer the unfolding social scenario (e.g. a fight, or a person helping another) but it is difficult to create a large number of varied artificial conversations that reflect the situation. On the other hand, subtasks such as extracting those sentences that address some specific topic, or (even simpler) extracting the first sentence of the input are simple enough to facilitate creating data points programatically. Note that while copying the first sentences might seem like a trivial or uninteresting pretraining task, it can be very useful. For example, in news summarization datasets the lead-3 baseline (copying over first 3 sentences as the summary) works very well (Brandow et al., 1995; Grenander et al., 2019).

Based on this analysis, we developed 21 elementary tasks, including copying specific content, performing numerical operations, and more. See Table 1 for full details on the slate of tasks.

Generating artificial summaries To create an input-summary pair using an elementary task from Table 1, we first create a base document and then (when required by the task) modify it by adding the requisite keywords. For example, *CopyKwdOneSentence* uses a keyword to mark the sentence to copy. The keywords added for tasks are also meaningless like *keyword1, keyword2*. Then the corresponding elementary operation is applied to generate the summary from this modified input.

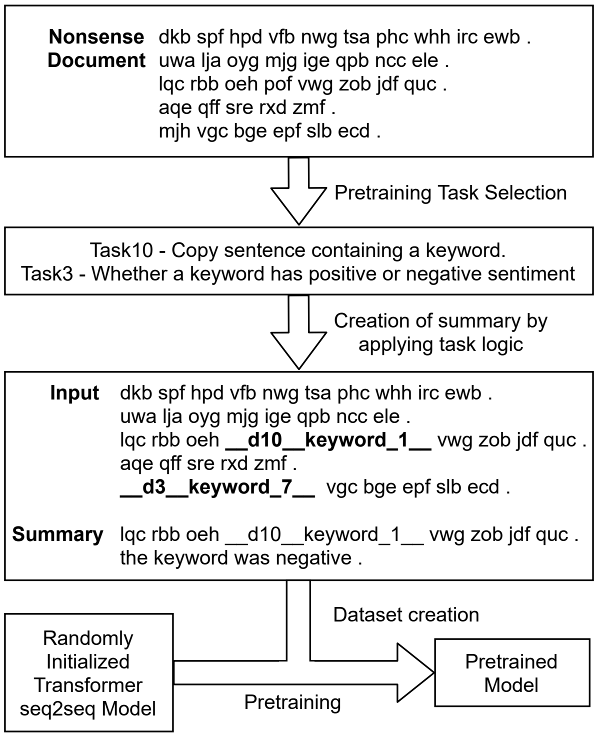

Figure 1: Procedure to create pretraining dataset using the nonsense corpus and our proposed pretraining tasks

The pretraining dataset that we create involves multiple elementary operations in each input-summary pair. To create the input-summary pair from a nonsense document, we first sample 3 elementary tasks and sequentially modify the input as needed by each task. Then, we generate the summary sentence(s) as required for each elementary task and concatenate them to constitute the overall summary. Here, the different keywords added to the input signal to the model which tasks are required to generate the summary. The procedure is illustrated in Figure 1.

6 Summarization Benchmarks

We fine-tune and evaluate our models on 4 downstream summarization benchmarks.

CNN-Dailymail-10K (See et al., 2017) Contains news articles and summaries from CNN and Dailymail websites. We use only $10k$ instances for

Elementary subtask	Description
CheckKeyword	Check if the input has a special keyword or not.
ClassifyKeyword	Output the category of keyword occurring in the input
MajorityKeyword	Out of two given keywords, find which one occurs more number of times
CopyFirstSentence	Copy first sentence
CopyBulleted	Copy over a bullet point (sentence starting with a bullet marker).
CopyQuoted	Copy text within quotes
CopyLastSentence	Copy last sentence
CopyKwdOneSentence	Copy the sentence that contains a keyword
CopyKwdMultipleSentInOrder	Copy all sentences containing any keyword in their order of appearance.
CopyKwdMultipleSentSorted	Copy all sentences containing any keyword, sorted by the keywords
CopyKwdMultipleSentShuffled	Copy all sentences containing keywords in any order.
ReplaceClassKeyword	Replace an object's mention with its category (e.g. apple $\rightarrow$ fruit)
CompareNumbers	Given two numbers in the text, say which one is bigger
SumOfNumbers	Sum all numbers in the input
ThresholdNumber	Check if a number in the input is above a threshold
LargestNumber	Find out largest of one or more numbers in the input.
TruncateSentence	Copy a sentence but only till the cutoff keyword is encountered
BreakClauses	Break a single sentence into multiple ones containing one clause each
JoinClauses	Join clauses from multiple sentences to make one longer sentence
ParaphraseWords	Copy a sentence while replacing its keywords with one of its synonyms
TopicSegregation	Copy sentences containing keywords from different classes into separate sections

Table 1: 21 extracted elementary summarization subtasks and their descriptions (detailed version is in Appendix)

training (randomly sampled from the training set) so that the impact of pretraining is more visible. However, we still evaluate the fine-tuned model on the full test set.

XSum-10K (Narayan et al., 2018) Also a news summarization dataset. Again, we train on a random subset of $10k$ instances from the training set.

Rottentomatoes (Wang and Ling, 2016) This dataset concerns summarizing critical reviews of movies found on the website `rottentomatoes.com`.

Rotowire (Wiseman et al., 2017) Here, the task is to process the box-score of a basketball game (often requiring numerical reasoning) to create a post-game summary.

7 Experiments and Results

First, we pretrain the transformer-based sequence-to-sequence architecture used by the T5 model (Raffel et al., 2020), on different corpora, each containing $100k$ input-summary pairs to get different pretrained models. We use the T5-small architecture in all experiments. Next, we fine-tune each model on the downstream tasks and measure performance via ROUGE score (Table 2). We also present the models' performance on next token prediction in summaries using accuracy and log-likelihood in the Appendix (Table 6). To frame the comparison, we include the performance of the official **T5** model and of a randomly initialized model using the same architecture (**T5-RI**) .

Pretraining with either our proposed pretraining tasks (**OurTasks**), or STEP tasks (**STEPTasks**) performs much better than random initialization, even when using nonsense data. For all summarization benchmarks except RottenTomatoes, the performance remained comparable when we used real upstream data from Wikipedia to create the pretraining datasets. This suggests that for some summarization benchmarks, there might be little or no additional benefit provided by using real world pretraining text.

Looking at individual STEPTasks, NSG has no training signal since the output is completely independent of the input, but surprisingly it leads to improvements in Rotowire benchmark. SR and MDG performed much better than NSG on CNN-DM and XSum, likely because they involve copying sentences/unmasked tokens from the input. We created *adjusted* versions of these pretraining datasets, where there was no copying needed and it led to a drop in performance on both pretraining tasks, bringing it close to **T5-RI** for CNN-DM and XSum. In SR-adjusted, the task is to output only the numerical order in which sentences should be copied (ver-

Model	CNN-DM-10K			XSum-10K			Rotten Tomatoes			Rotowire		
	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL
T5-OffShelf	39.38	18.08	27.71	29.18	8.69	22.62	24.73	9.00	19.64	37.50	12.85	19.85
T5-RI	9.86	1.06	7.49	15.49	2.48	12.76	10.17	0.41	8.66	4.02	0.72	3.68
Nonsense Upstream Corpus												
T5-OurTasks	35.23	14.77	24.03	20.36	4.15	16.23	15.72	2.06	12.51	39.10	11.81	19.94
T5-STEPTasks	35.78	14.98	23.60	21.49	4.56	16.78	13.22	0.88	10.83	29.82	7.45	16.74
T5-STEPTask-NSG	9.20	0.80	7.19	15.78	2.24	12.44	12.31	0.71	10.60	33.65	7.60	17.90
T5-STEPTask-SR	28.63	10.67	20.35	21.47	4.70	16.62	10.89	0.51	9.18	25.68	5.39	15.29
T5-STEPTask-SR-adjusted	7.24	0.63	5.69	15.04	2.00	12.12	11.18	0.46	9.51	20.00	2.74	12.08
T5-STEPTask-MDG	34.50	14.45	23.77	20.76	4.13	16.45	11.78	0.70	9.89	36.22	10.53	18.73
T5-STEPTask-MDG-adjusted	10.15	0.93	7.78	16.12	2.20	13.09	15.07	1.38	11.69	20.39	3.77	11.97
Real Upstream Corpus												
T5-OurTasks	34.06	13.88	23.21	22.27	5.09	17.60	19.16	5.26	15.65	38.57	11.89	19.68
T5-STEPTasks	32.04	12.93	22.55	23.37	5.68	18.42	20.89	6.29	17.05	37.63	10.89	19.57
PG Models Randomly Initialized vs Pretrained (Nonsense Upstream Corpus)												
PG-RI	29.68	11.75	21.82	17.66	3.57	14.62	19.63	6.43	16.62	30.61	8.66	17.74
PG-OurTasks	29.82	11.78	21.91	16.81	3.43	13.95	19.02	6.57	16.38	26.94	6.81	16.77
PG-STEPTasks	29.44	11.74	21.67	17.65	3.54	14.55	17.70	5.89	15.34	31.16	8.49	17.85

Table 2: Rouge scores achieved by different models on four summarization benchmarks.

sus actually generating the full output). In MDG-adjusted, the task is to only output the masked-out tokens (versus outputting the entire document, including unmasked tokens).

A randomly initialized pointer-generator model (See et al., 2017) (**PG-RI**) performs far better than a randomly initialized T5 model. However, T5-architecture models pretrained on nonsense text were able to outperform pointer-generator on 3 out of 4 benchmarks, suggesting that transformer models pretrained on nonsense text can be a better choice than using non-pretrained LSTM based models. Interestingly, pretraining the PG model on either **OurTasks** or **STEPTasks** did not lead to any additional improvement.

Models pretrained separately on each task from **OurTasks** exhibit strong differences in their performance on CNN-Dailymail-10K benchmark (Table 3). Models pretrained on *TopicSegregation* and *CopyKwdMultipleSent-Shuffled* outperform others significantly. The two worst performing models were pretrained on *CompareNumbers* and *SumOfNumbers*, and these models were unable to perform any better than random guessing on the pretraining task itself. By contrast, most other pretrained models were able to solve their pretraining task correctly more than 99% of times (see Table 7 in Appendix for full details).

Pretraining task	R1	R2	Pr%
TopicSegregation	23.04	7.79	99.90
CopyKwdMultipleSent-Shuffled	23.34	5.46	99.66
TruncateSentence	17.07	2.50	1.00
LargestNumber	6.52	0.58	99.88
SumOfNumbers	5.03	0.40	25.06
CompareNumbers	1.89	0.04	48.88

Table 3: The 3 best and worst performing pretraining tasks according to performance of their pretrained models on CNN-Dailymail-10K (R1,R2), and their accuracy on the pretraining task (Pr%).

8 Conclusion

This paper demonstrated that transformer models pretrained on randomly generated nonsense data deliver remarkable performance gains across multiple summarization tasks, compared to their randomly initialized version. This suggests that a substantial part of the observed benefits of pretraining can not be attributed to knowledge transfer. To investigate whether the design of pretraining task itself plays a significant role and can lead to further performance gains, we explored summarization datasets to prepare a battery of tasks found useful in creating summaries. But these pretraining tasks performed comparably to more generic pretraining tasks used in literature. Our work suggests that understanding pretraining may have more to do with poorly-understood aspects of how initialization influences optimization than with knowledge transfer.

References

Miltiadis Allamanis, Hao Peng, and Charles Sutton. 2016. A convolutional attention network for extreme summarization of source code. In *International conference on machine learning*, pages 2091–2100. PMLR.

Shikha Bordia and Samuel Bowman. 2019. Identifying and reducing gender bias in word-level language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 7–15.

Ronald Brandow, Karl Mitze, and Lisa F Rau. 1995. Automatic condensation of electronic publications by sentence selection. *Information Processing & Management*, 31(5):675–685.

Cheng-Han Chiang and Hung-yi Lee. 2020. Pre-training a language model without human language. *arXiv preprint arXiv:2012.11995*.

Eric Chu and Peter Liu. 2019. Meansum: a neural model for unsupervised multi-document abstractive summarization. In *International Conference on Machine Learning*, pages 1223–1232. PMLR.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. 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)*, pages 615–621.

Thomas Davidson, Debasmita Bhattacharya, and Ingmar Weber. 2019. Racial bias in hate speech and abusive language detection datasets. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 25–35.

Shima Gerani, Giuseppe Carenini, and Raymond T Ng. 2019. Modeling content and structure for abstractive review summarization. *Computer Speech & Language*, 53:302–331.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *EMNLP-IJCNLP 2019*, page 70.

Matt Grenander, Yue Dong, Jackie Chi Kit Cheung, and Annie Louis. 2019. Countering the effects of lead bias in news summarization via multi-stage training and auxiliary losses. In *EMNLP-IJCNLP*, pages 6021–6026.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Chin-Yew Lin and Eduard Hovy. 2002. Manual and automatic evaluation of summaries. In *Proceedings of the ACL-02 Workshop on Automatic Summarization-Volume 4*, pages 45–51.

Yongjie Lina, Yi Chern Tana, and Robert Frankb. 2019. Open sesame: Getting inside bert's linguistic knowledge. *ACL 2019*, page 241.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of EMNLP-IJCNLP 2019*, pages 3721–3731.

Hartmut Maennel, Ibrahim Alabdulmohsin, Ilya Tolstikhin, Robert JN Baldock, Olivier Bousquet, Sylvain Gelly, and Daniel Keysers. 2020. What do neural networks learn when trained with random labels?

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2020. What is being transferred in transfer learning? In *Advanes in Neural Information Processing Systems (NeurIPS)*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206.

Isabel Papadimitriou and Dan Jurafsky. 2020. Learning music helps you read: Using transfer to study linguistic structure in language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6829–6839.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6908–6915.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2401–2410.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. 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)*, pages 1073–1083.

Eva Sharma, Chen Li, and Lu Wang. 2019. Bigpatent: A large-scale dataset for abstractive and coherent summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2204–2213.

Koustuv Sinha, Robin Jia, Dieuwke Hupkes, Joelle Pineau, Adina Williams, and Douwe Kiela. 2021. Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. *arXiv preprint arXiv:2104.06644*.

Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. A graph-to-sequence model for amr-to-text generation. *arXiv preprint arXiv:1805.02473*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *ACL*, pages 4593–4601.

Pavlos Vougiouklis, Hady Elsahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon Hare, and Elena Simperl. 2018. Neural wikipedian: Generating textual summaries from knowledge base triples. *Journal of Web Semantics*, 52:1–15.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1395–1405.

Lu Wang and Wang Ling. 2016. Neural network-based abstract generation for opinions and arguments. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 47–57.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32:5753–5763.

Michihiro Yasunaga, Jungo Kasai, Rui Zhang, Alexander R Fabbri, Irene Li, Dan Friedman, and Dragomir R Radev. 2019. Scisummnet: A large annotated corpus and content-impact models for scientific paper summarization with citation networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7386–7393.

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

Yanyan Zou, Xingxing Zhang, Wei Lu, Furu Wei, and Ming Zhou. 2020. Pre-training for abstractive document summarization by reinstating source text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3646–3660.

A Appendix

Hyperparameters We use the T5-Small architecure with 60.5 million parameters as our transformer-based model. The models are all trained using the BertAdam optimizer with a learning rate of 10^{-4}. For the pointer-generator model, the token embedding size is 128, its encoder is a bidirectional LSTM with hidden size 256 the decoder is a unidirectional LSTM of the same size. The entire model had 4.4 million parameters. For a fair comparision, we use wordpiece tokenization with all models with the same tokenizer and vocabulary as used by the standard T5 model. The validation metric used in all experiments was accuracy on the next-token prediction on the summaries. A patience value of 5 epochs was used for early stopping.

For CNN-Daiymail dataset, we truncated the input and output lengths according to Zou et al. (2020) (Table 5). We use the same lengths for the XSum dataset as well . For the Rotowire and Rottentomatoes dataset, the input and output lengths were much longer and even with a batch size of 1, we had to truncate them to values that allowed us to accommodate training with the available GPU memory (32GB). While decoding, we used beam search with beam size 4, and set the minimum and maximum decoding lengths to the 5 and 95 percentile of their observed distribution.

Computing infrastructure Most experiments were carried out on 8 Nvidia V100 GPUs with 32 GB of memory. Some experiments with CNN-Dailymail and XSum datasets were carried out on 4 Nvidia RTX2080Ti GPUs with 11GB of memory.

Exclusions from ensemble of our tasks When creating artificial summaries requires using multiple of our proposed elementary tasks, the different keywords added to the input signal to the model which tasks are required for it. Three of our proposed tasks do not always involve keyword addition— *CopyFirstSentence, CopyLastSentence, CheckKeyword*. Hence we exclude them when creating the pretraining corpus with our ensemble of tasks. We also exclude the *SumOfNumbers* and *CompareNumbers* tasks because they could not be learnt even in isolation by a randomly initialized T5 model training on 100k datapoints.

Details of dataset splits For the Rotowire and RottenTomatoes datasets, we use the standard training, validation and test splits with sizes shown in Table 4. For the CNN-Dailymail and XSum datasets, we use the standard test splits, but reduce the training and validation set sizes to 10k and 1k respectively by uniformly subsampling from the standard full dataset splits.

Evaluation metrics We measure the quality of generated summaries using ROUGE scores (Lin and Hovy, 2002) which measure n-gram overlap between a generated and reference summary to assess its quality. We use the ROUGE-1,2 and L variants of this metric which measure overlap in unigrams, bigrams and longest common subsequence respectively. We also present the average performance of models at predicting the next token of a summary given all the ground truth past tokens (Table 6). To measure this, we use the accuracy and the negative-log-likelihood metrics which are standard for multi-class classification. We average these metrics across different decoding timesteps of summary generation, and then average it again across all the summaries in the test set.

	CNN-DM-10K	XSum-10K	RottenTomatoes	Rotowire
Train	10000	10000	2458	3398
Validation	1000	1000	536	727
Test	11490	11333	737	728

Table 4: Sizes for Train, validation and test splits for all datasets

	CNN-DM-10K	XSum-10K	RottenTomatoes	Rotowire
max source length	512	512	6000	5160
max target length	256	256	∞	815
batch size	16	16	1	1
max decode length	148	42	52	815
min decode length	44	18	16	223

Table 5: Hyperparameters used for fine-tuning models on the 4 datasets

Experiment	CNN-DM-10K		XSum-10K		Rottentomatoes		Rotowire	
	Acc	NLL	Acc	NLL	Acc	NLL	Acc	NLL
T5-OffShelf	65.15	1.71	53.68	2.34	51.78	2.77	68.04	1.50
T5-RandomInit	29.78	4.92	32.60	4.75	24.75	5.36	48.30	2.61
Nonsense Upstream Corpus								
T5-OurTasks	54.74	3.18	38.98	4.27	33.42	5.08	63.59	1.78
T5-STEPTasks	54.71	3.18	39.47	4.21	28.65	5.13	58.89	1.99
Real Upstream Corpus								
T5-OurTasks	54.87	2.93	41.21	3.76	39.64	4.12	64.02	1.78
T5-STEPTasks	57.91	2.46	46.83	3.08	45.34	3.43	64.08	1.63
PG Models Randomly Initialized vs Pretrained (Nonsense Upstream Corpus)								
PG-RandomInit	51.14	2.91	33.05	4.14	33.35	4.37	59.12	1.92
PG-OurTasks	51.70	2.89	33.80	4.14	34.40	4.29	59.30	1.92
PG-STEPTasks	51.79	2.88	34.13	4.14	35.06	4.21	59.00	1.94

Table 6: Accuracy (Acc) and negative log likelihood (NLL) for next token prediction on summaries

Pretraining task	R1	R2	RL	Pr%
CopyKwdMultipleSent-Shuffled	**23.34**	5.46	15.41	99.66
TopicSegregation	23.04	**7.79**	**16.52**	99.88
TruncateSentence	17.07	2.50	11.81	100.00
CopyQuoted	11.03	1.32	8.32	99.82
BreakClauses	10.46	1.18	7.95	99.80
CopyKwdMultipleSent-InOrder	10.14	1.14	7.70	99.84
ReplaceClassKeyword	9.70	0.95	7.36	99.98
ParaphraseWords	9.70	0.99	7.42	99.98
CopyKwdOneSentence	9.45	1.06	7.23	99.90
CopyFirstSentence	9.28	1.08	7.22	99.88
CopyBulleted	9.01	1.00	6.88	99.58
CopyKwdMultipleSent-Sorted	8.48	0.83	6.59	99.68
MajorityKeyword	8.45	0.85	6.49	100.00
ThresholdNumber	7.83	0.77	6.05	100.00
CheckKeyword	7.79	0.77	5.94	100.00
CopyLastSentence	7.78	0.72	6.12	98.40
JoinClauses	7.72	0.81	6.09	98.82
ClassifyKeyword	6.80	0.62	5.34	100.00
LargestNumber	6.52	0.58	5.14	99.88
SumOfNumbers	5.03	0.40	4.14	25.06
CompareNumbers	1.89	0.04	1.75	48.88

Table 7: For different models pretrained on one individual task each, their performance on CNN-Dailymail-10K in terms of ROUGE (R1,R2,RL), and their accuracy in percentage on the pretraining task (Pr%)

Elementary subtask	Description
CheckKeyword	Check if the input has a special keyword or not.
ClassifyKeyword	Input contains 1 of 10 special keywords - 5 or them are positive and 5 of them are negative adjectives. Task is to tell whether mentioned adjective was positive or negative
MajorityKeyword	Out of two given keywords, find which one occurs more number of times
CopyFirstSentence	Copy first sentence
CopyBulleted	Exactly one sentence is a bullet point and starts with the bullet marker. You have to copy over that sentence without copying the marker.
CopyQuoted	Copy text within quotes
CopyLastSentence	Copy last sentence
CopyKwdOneSent	Copy single sentence containing one of many special defined keywords
CopyKwdMultipleSentInOrder	Copy all sentences containing any special keyword in the same order as they appear in text.
CopyKwdMultipleSentSorted	Copy all sentences containing keywords but sort them according to the canonical ordering of keywords
CopyKwdMultipleSentShuffled	Copy all sentences containing keywords in any order. The sentences in ground truth may be any possible order.
ReplaceClassKeyword	There exist many keywords, each belonging to one of 3 classes. You have to mention the class of the mentioned keyword
CompareNumbers	Given two numbers in the text, say which one is bigger
SumOfNumbers	Sum numbers
ThresholdNumber	The input contains a number between 0 and 100. You have to say if the number was above or equal to the threshold of 50 of lower than it
LargestNumber	Find out largest of one or more numbers in the input.
TruncateSentence	Copy a sentence but only till the cutoff keyword is encountered
BreakClauses	Break a single sentence into multiple ones containing one clause each
JoinClauses	Join clauses from multiple sentences to make one longer sentence
ParaphraseWords	Copy the sentence containing one of pre-specified special keywords. But replace the keyword with any of its multiple synonyms. The j^{th} synonym of i^{th} keyword src_i is given by $target_{ij}$
TopicSegregation	Copy all sentences containing keywords belonging to different classes but put them in corresponding sections (each class gets a separate section, which can be empty too, sections always occur in sorted order)

Table 8: 21 extracted elementary summarization subtasks and their descriptions

Domain	Dataset name	Paper using the dataset
News	CNN-Dailymail	See et al. (2017)
	NYT	Paulus et al. (2018)
	Gigaword	Paulus et al. (2018)
	XSUM	Liu and Lapata (2019)
	Newsroom	Zhang et al. (2020)
Code	Code to Documentation dataset	Iyer et al. (2016)
	Git diff to commit-message dataset	Allamanis et al. (2016)
Scientific Paper	Arxiv	Cohan et al. (2018)
	Pubmed	Cohan et al. (2018)
	ScisummNet	Yasunaga et al. (2019)
Patent	BigPatent	Sharma et al. (2019)
Instructional guides	Wikihow	Zhang et al. (2020)
Social media post	Reddit-TIFU	Zhang et al. (2020)
Email	AESLC	Zhang et al. (2020)
Bills	BillSum	Zhang et al. (2020)
Reviews	Amazon reviews	Gerani et al. (2019)
	Yelp reviews	Chu and Liu (2019)
	CNET reviews	Gerani et al. (2019)
KeyValue Attributes	Wikibio	Lebret et al. (2016)
	E2E dataset	Novikova et al. (2017)
Knowledge Graphs	DBPedia triples to Wikipedia	Vougiouklis et al. (2018)
	AMR to sentence dataset	Song et al. (2018)
	Agenda	Koncel-Kedziorski et al. (2019)
	WebNLG	Moryossef et al. (2019)
Numerical Table	Rotowire box-score	Puduppully et al. (2019)
Miscellaneous webpages	Wikisum	Liu et al. (2018)
Conversations	SamSum	Gliwa et al. (2019)
	AMI	Wang and Cardie (2013)

Table 9: Existing summarization datasets in various domains, along with corresponding papers that use them and came up during the search procedure to characterize elementary tasks in summarization