Improving the Robustness of Conditional Language Models by Detecting and Removing Input Noise

Kundan Krishna[1*] **Yao Zhao**[2] **Jie Ren**[2] **Balaji Lakshminarayanan**[2]
Jiaming Luo[2] **Mohammad Saleh**[2] **Peter J Liu**[2*]
[1]Carnegie Mellon University, work done while at Google Research
[2]Google Research
[*]Correspondence to: kundank@andrew.cmu.edu, peterjliu@google.com

Abstract

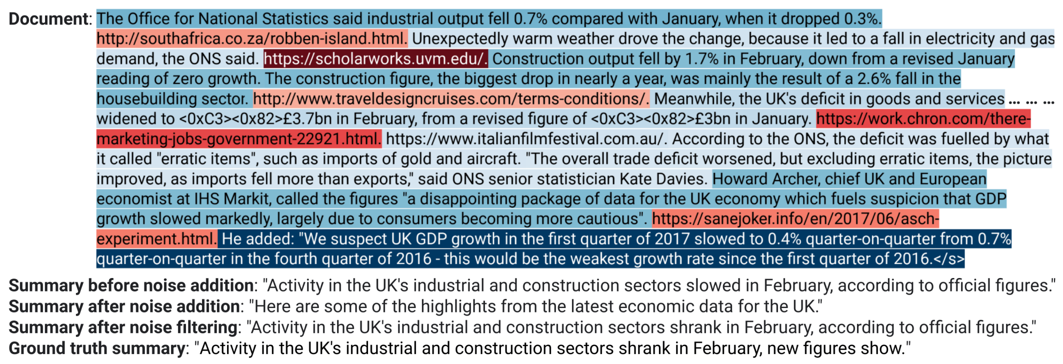
The evaluation of conditional language modeling tasks such as abstractive summarization typically uses test data that is identically distributed as training. In real-world practice, documents to be summarized may contain input noise caused by text extraction artifacts or data pipeline bugs. The robustness of model performance under distribution shift caused by such noise is relatively under-studied. We present a large empirical study quantifying the sometimes severe loss in performance – up to 12 ROUGE-1 points – from different types of input noise for a range of datasets and model sizes. We then propose a light-weight method for detecting and removing such noise in the input during model inference without requiring any extra training or auxiliary models, which effectively mitigates the loss in performance, recovering up to 11 ROUGE-1 points.

1 Introduction

Despite rapid progress in conditional language modeling and abstractive summarization in particular in recent years [9, 17, 21], virtually all works have tested models using identically distributed data as training, and little attention has gone into studying their robustness to input distribution shift caused by input noise. Different forms of data which are summarized can often contain noise — chats may contain URLs or unseen emojis, news articles on the web may have embedded ads or tweets, and scanned documents may contain OCR errors [5]. However, the impact of different kinds of noise on modern abstractive summarization systems, and how to accurately detect and remove that noise, remains largely unknown. In contrast, much work has been done for anomaly segmentation of images, i.e. detecting and localizing unknown objects in images [3, 1, 12], suggesting a gap between the study on vision and language in fine-grained input noise detection.

In this work, we study how noise in the input affects the output generated by conditional language models, and propose a method to detect and remove it without relying on an external model. We use summarization as a case study. We inject 4 commonly observed types of noise to 4 abstractive summarization datasets with diverse styles [13, 8, 2, 19], and quantify the drop in aggregate metrics for the output summaries (Section 2). We also study how the quality of generated summaries varies with factors such as the amount of noise and size of the models. For our experiments, we use PEGASUS [21] models - Transformer-based pre-trained models which deliver competitive performance across abstractive summarization benchmarks.

We experiment with methods to detect and remove noisy spans in the input, and show that we can recover a large fraction of the drop in output quality resulting from the addition of noise (Section 3). Our approach for detecting noisy spans is based on variations of the out-of-distribution (OOD) detection techniques proposed by Ren et al. [18] — *RMD Input Score*, which uses the embeddings computed by the summarization model's encoder without requiring any additional model or training. Figure 1 shows our method's impact on a sample noisy document.


Document: The Office for National Statistics said industrial output fell 0.7% compared with January, when it dropped 0.3%. http://southafrica.co.za/robben-island.html. Unexpectedly warm weather drove the change, because it led to a fall in electricity and gas demand, the ONS said. https://scholarworks.uvm.edu/. Construction output fell by 1.7% in February, down from a revised January reading of zero growth. The construction figure, the biggest drop in nearly a year, was mainly the result of a 2.6% fall in the housebuilding sector. http://www.traveldesigncruises.com/terms-conditions/. Meanwhile, the UK's deficit in goods and services ... ... widened to <0xC3><0x82>£3.7bn in February, from a revised figure of <0xC3><0x82>£3bn in January. https://work.chron.com/there-marketing-jobs-government-22921.html. https://www.italianfilmfestival.com.au/. According to the ONS, the deficit was fuelled by what it called "erratic items", such as imports of gold and aircraft. "The overall trade deficit worsened, but excluding erratic items, the picture improved, as imports fell more than exports," said ONS senior statistician Kate Davies. Howard Archer, chief UK and European economist at IHS Markit, called the figures "a disappointing package of data for the UK economy which fuels suspicion that GDP growth slowed markedly, largely due to consumers becoming more cautious". https://sanejoker.info/en/2017/06/asch-experiment.html. He added: "We suspect UK GDP growth in the first quarter of 2017 slowed to 0.4% quarter-on-quarter from 0.7% quarter-on-quarter in the fourth quarter of 2016 - this would be the weakest growth rate since the first quarter of 2016.</s>

Summary before noise addition: "Activity in the UK's industrial and construction sectors slowed in February, according to official figures."
Summary after noise addition: "Here are some of the highlights from the latest economic data for the UK."
Summary after noise filtering: "Activity in the UK's industrial and construction sectors shrank in February, according to official figures."
Ground truth summary: "Activity in the UK's industrial and construction sectors shrank in February, new figures show."

Figure 1: Effect of noise addition and filtering on the model generated summary for a sample document. Random URLs are injected to the original document as noise. The color indicates the value of our proposed OOD score for a text span — red represents positive and blue represents negative OOD scores, with saturation proportional to the magnitude. Removing the detected noisy parts from input and feeding to summarization model results in a summary closer to the ground truth.

A summary of our contributions:

- We quantify the impact of various kinds of noise on state-of-the-art summarization models, demostrating drops in output quality measuring as large as 12 ROUGE-1 points.
- We show this noise can be detected using our proposed out-of-distribution methods, without ever seeing it in training. Much of the performance drop can be recovered (up to around 11 ROUGE-1), improving robustness and safety for real-world model deployment.

2 Impact of noise addition

We inject noisy text spans in between sentences of the clean articles. The insert position of each noisy text span is sampled independently and uniformly at random (see Figure A.1 in Appendix for an example). Overall, we consider the following choices of a noisy text span:

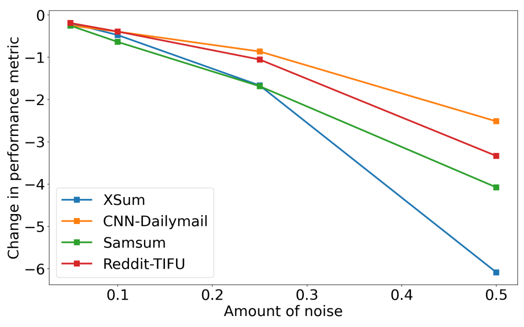
- **Code** - a random line of code from a corpus of Python programs [4]
- **Emoji** - randomly sampled emojis taken from the version 15 release on `unicode.org`
- **Randomsent** - a random sentence from the first 1% of validation set of the Colossal Common Crawl Corpus(C4) [17].
- **URL** - a random URL from the first 1% of validation set of the C4 corpus.

We experiment with different amounts of noise added to the input which is treated as a hyper-parameter. We measure the amount of noise in terms of the number of noisy tokens added to the input divided by the total number of tokens in the input after noise addition. We experiment with 4 different datasets — XSUM [13], CNN/DailyMail [19], SAMSum [2] and RedditTIFU-long [7]. Our datasets span a variety of domains, where the first two datasets deal with summarizing news articles, and the remaining two consider summarizing conversations and social media posts respectively. For all experiments with each summarization dataset, we use PEGASUS models [21] finetuned on that dataset. We evaluate the performance of models using ROUGE scores [10] of the corresponding summaries generated by the them.

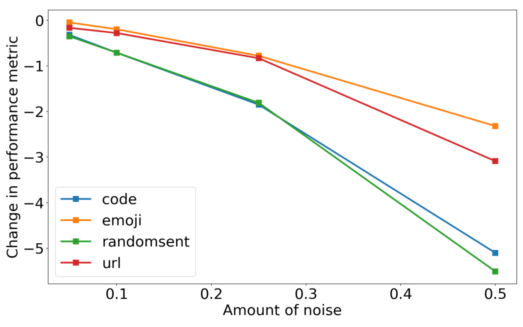
Effect of noise amount: We compare four different levels of noise, 5%, 10%, 25%, and 50% (50% means the amount of noise tokens equals to the amount of the clean tokens.). As shown in Figure 2, we see a near monotonic decrease in output quality as more noise is added to the data. In Figure 2a, we group it by datasets while averaging across model sizes and noise types. This reveals that some datasets are more robust to noise than others (e.g. CNN/DailyMail is most robust), and the relative trends in performance drops remain similar across different noise amounts. In Figure 2b, we group the performance drops by noise types while averaging across datasets and model sizes. We see a clear gap between the drops for Code and Randomsent vs Emoji and URL, with the gap widening as the noise amount is increased.

Effect of noise type: In general, we see the models are more robust to URLs and emojis, and less robust to Randomsent and Code noise types as demonstrated by performance drops (averaged across model sizes) shown in Figure 2c. We suspect that some of the this could be due to the presence of URLs and emojis in the training dataset itself, due to which the model may have learned to be robust to those noise types. Models trained on different datasets have varying sensitivity to different kinds of noises (Figure 2c). For example, SAMSum is notoriously susceptible to Randomsent noise, leading to a drop of about 10 Rouge-1 points averaged across model sizes, whereas for CNN/DailyMail Code is the most harmful type of noise.

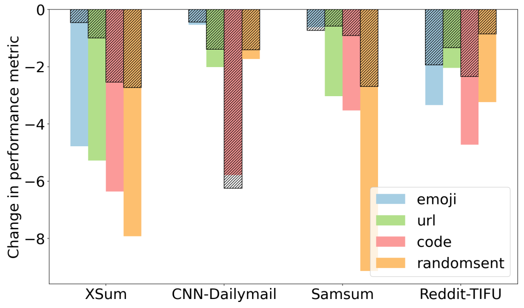
Effect of model size: We compare PEGASUS models of 3 different sizes (number of parameters) — Small (50M), Base (200M), and Large (500M). As shown by performance drops (averaged over noise types) in Figure 2d, one might expect larger models to be less susceptible to noise, but it does not seem to be the case in general and simply scaling up models may not solve robustness. In some cases, large models can still suffer loss of over 10 ROUGE-1 points with addition of noise (see Table A.1 in Appendix).



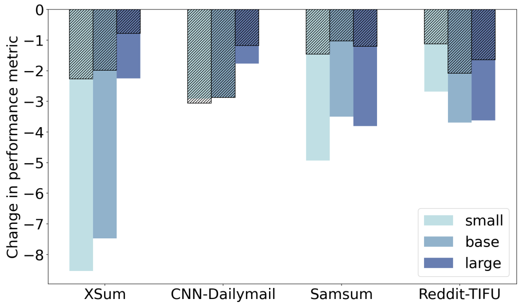
(a) Effect of noise amount by dataset



(b) Effect of noise amount by noise type



(c) Effect of noise type by dataset



(d) Effect of model size by dataset

Figure 2: Change in output quality upon addition of noise to inputs, while varying different factors — noise amount in (a) and (b), noise type in (c), and model size in (d). In (c) and (d) we also show the quality after noise removal (the shaded area). Quality is measured as the geometric mean of ROUGE-1/2/L scores and averaged over the non-varying factors. We set noise amount to 0.5 in (c) and (d).

3 Noise detection and quality recovery

3.1 Noise detection

Ren et al. [18] studied various methods for detecting OOD inputs for conditional language generation tasks, including summarization. They showed that the proposed embedding-based OOD detection method *Relative Mahalanobis distance* (RMD) worked well. Specifically, given an input sequence $\boldsymbol{x} = x_1 \dots x_t$, the method obtains the input embedding $\boldsymbol{z} = \frac{1}{t}\Sigma_i \boldsymbol{h}_i$ by averaging the encoder's final-layer hidden state vectors $\boldsymbol{h}_i$ corresponding to the input sequence token x_i. The OOD score is defined as the difference between two *Mahalanobis distances* (MD),

$$S(\boldsymbol{x}) := \mathrm{RMD}(\boldsymbol{z}) := \mathrm{MD}_{in}(\boldsymbol{z}) - \mathrm{MD}_0(\boldsymbol{z}), \tag{1}$$

where $\mathrm{MD}_{in}(\boldsymbol{z}) = (\boldsymbol{z} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{z} - \boldsymbol{\mu})$ measures the distance from $\boldsymbol{z}$ to the fitted in-domain Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and $\mathrm{MD}_0(\boldsymbol{z}) = (\boldsymbol{z} - \boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1}(\boldsymbol{z} - \boldsymbol{\mu}_0)$ measures the distance to the fitted background Gaussian $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. The in-domain distribution is fitted using the in-domain training data, and the background distribution is fitted using the same number of examples from `C4` [16] which is represents a large and broad set of domains. Instead of computing OOD score for the entire input sequence as in [18], in this work, we focus on detecting smaller sub-parts of OOD noise within the sequence. We propose three variants:

Leaveout-Sentence (LO-Sent) In this case, we compute the OOD scores of the input with and without a sentence in it. The negative of the change in the OOD score after removing the sentence denotes the OOD score of that sentence. Intuitively, if removing the sentence decreases the overall OOD score, that sentence is assigned a positive OOD score and vice-versa.

$$S_{\text{LO-Sent}}(\boldsymbol{x}_{i:j}) = S(\boldsymbol{x}_{1:t}) - S(\boldsymbol{x}_{1:i;j:t}) \tag{2}$$

Leaveout-Token (LO-Tok) This is very similar to the previous method LO-Sent except that instead of removing a sentence, we remove a token at a time and hence get OOD scores for each token,

$$S_{\text{LO-Tok}}(x_i) = S(\boldsymbol{x}_{1:t}) - S(\boldsymbol{x}_{1:i;(i+1):t}). \tag{3}$$

Sentencewise (Sent) Instead of computing the score based on embeddings averaged over the tokens in the whole document, we fit Gaussian distributions at the sentence level by averaging the token embeddings in a sentence $\boldsymbol{z}_{i:j} = \frac{1}{j-i+1}\sum_{k=i}^{j} \boldsymbol{h}_k$. We use the sentence embeddings from in-domain data and `C4` data to fit the two Gaussian distributions, $\mathcal{N}(\boldsymbol{\mu}^{\text{sent}}, \boldsymbol{\Sigma}^{\text{sent}})$ and $\mathcal{N}(\boldsymbol{\mu}_0^{\text{sent}}, \boldsymbol{\Sigma}_0^{\text{sent}})$.

$$S_{\text{sent}}(\boldsymbol{x}_{i:j}) = \text{RMD}_{\text{sent}}(\boldsymbol{z}_{i:j}) = \text{MD}_{in}^{\text{sent}}(\boldsymbol{z}_{i:j}) - \text{MD}_0^{\text{sent}}(\boldsymbol{z}_{i:j}) \tag{4}$$

where $\text{MD}_{in}^{\text{sent}}$ and $\text{MD}_0^{\text{sent}}$ are MDs to $\mathcal{N}(\boldsymbol{\mu}^{\text{sent}}, \boldsymbol{\Sigma}^{\text{sent}})$ and $\mathcal{N}(\boldsymbol{\mu}_0^{\text{sent}}, \boldsymbol{\Sigma}_0^{\text{sent}})$ respectively.

To calculate performance of models at noise detection, we compare the assigned OOD score for each token with its ground truth label and we compute the ROC AUC scores for comparison. For the two sentence level scores, $S_{\text{LO-Sent}}(\boldsymbol{x}_{i:j})$ and $S_{\text{sent}}(\boldsymbol{x}_{i:j})$, we assign each token's OOD score to be the sentence level OOD score for the sentence to which the token belongs. We compute evaluation metrics in two ways - (1) *per-example* basis where the AUC score is computed for each example and then they are all averaged across the dataset. (2) *overall* basis where all the predictions across the entire dataset are pooled together before computing a single AUC score. In general, the LO-Tok method performs the worst of the three methods, while Sent and LO-Sent perform comparably. We show the scores averaged across the 4 datasets in (Table 1). Sent performs better for Code and Randomsent and LO-Sent performs better for Emoji and URL noise types. For its simplicity, we use the Sent method for OOD detection in rest of the paper.

Table 1: Performance of different methods for noise detection aggregated across datasets (using the base model size and 0.5 noise amount)

Method	**Overall AUC**				**Per-example AUC**			
	Code	Emoji	Randomsent	URL	Code	Emoji	Randomsent	URL
LO-Tok	77.10	84.25	73.63	85.41	78.52	84.17	74.74	86.83
LO-Sent	88.04	88.83	85.43	95.66	89.46	87.94	87.00	96.08
Sent	89.37	82.73	90.65	90.64	91.70	82.80	93.83	93.64

3.2 Quality recovery after noise filtering

To remove noise from the input, we simply remove all sentences that have an OOD score greater than a threshold, and then evaluate how much output quality gets recovered after this. We set the threshold of OOD score for filtering to be the 99 percentile value of the OOD scores computed for sentences in the clean version of the dataset (without any noise). The chosen percentile is set to be this high to minimize false positives which can lead to removal of useful non-noisy information from the input. Since the threshold is computed using only the clean dataset and the model trained on that, we do not need any prior information about the noise (similar to OOD score computation).

We show the performance of noise filtering for different noise types, model sizes and datasets in Table 2. For succinctness, we show the geometric mean of the ROUGE-1,2 and L variants, and point the reader to the Appendix (Table A.1) for detailed results with individual variants of ROUGE. After noise filtering, we can recover a large part of the drop in ROUGE scores that occurred due to the added noise. In cases of large drop such as the Randomsent noise type with XSUM and SAMSum datasets, we can recover 4-6 and 6-7 points respectively depending on the model size (Table 2).

Table 2: ROUGE scores on clean input and changes when adding different kinds of noise, and after the noise is filtered out using Sent method (Noise amount: 0.5)

Model size	**Clean**	**Code**		**Emoji**		**Randomsent**		**URL**	
		Add	Filter	Add	Filter	Add	Filter	Add	Filter
				XSum					
Small	31.66	21.43	27.50	23.28	31.33	22.28	28.44	25.50	30.30
Base	35.18	27.64	32.01	30.03	34.49	26.28	32.32	26.87	33.97
Large	37.18	35.86	36.89	36.36	36.83	31.68	35.09	35.81	36.77
				CNN-Dailymail					
Small	31.96	25.27	23.37	31.24	31.46	30.01	30.38	29.69	30.39
Base	33.09	26.27	25.39	32.53	32.70	31.31	31.53	30.74	31.25
Large	33.44	29.60	30.99	33.11	33.02	31.97	32.36	32.03	32.67
				Samsum					
Small	37.96	33.00	36.80	36.83	36.73	28.11	35.18	34.17	37.31
Base	39.74	36.95	38.89	39.18	38.97	31.96	37.51	36.89	39.47
Large	41.63	38.80	40.91	41.46	41.42	31.85	38.58	39.19	40.81
				Reddit-TIFU					
Small	15.51	11.53	13.55	12.97	15.21	13.40	14.70	13.41	14.09
Base	17.54	12.16	14.55	13.33	14.42	14.18	16.62	15.71	16.23
Large	18.15	13.33	16.06	14.89	15.76	13.92	17.32	15.96	16.88

We also present aggregate trends of recovery of output quality using our filtering approach in Figure 2. We can see that we recover over half of the drop in the performance on 9 out of 16 combinations of datasets and noise types (Figure 2c), with the best performance observed on XSUM and SAMSum datasets and the worst on CNN/DailyMail. The method also succeeds in recovering performance across all 3 model sizes (Figure 2d).

4 Related Work

Research on the behavior of summarization models on noisy inputs is quite sparse. Jing et al. [5] investigated how the performance of extractive summarization models is impacted by noise due to OCR errors while summarizing scanned documents. More recently, Meechan-Maddon [11] studied the effect of noise in the form of ASR errors on abstractive summarization models based on convolutional neural networks. In contrast, we experiment with pre-trained Transformer models which are now preferred in popular use due to their superior performance [9, 21, 17], and address a wide variety of noise types and summarization datasets.

The effect on noisy inputs has been studied in NLP tasks other than summarization, such as machine translation [14] and question answering [15]. Multiple works across machine translation [6, 20], question answering [15] and summarization [5] have used synthetic noise to create noisy inputs. Similar to these works, we also create synthetic noisy inputs due to lack of a dataset with naturally occurring labeled noise. One significant distinction of our work is our noise detection/removal method works even without exposing the model to the noise in training, which is closer to practical scenarios where unknown types of noise can be encountered after a model is deployed.

5 Conclusion and Future Work

In this work, we quantified the impact that noisy inputs can have on the output quality of summarization models, for a variety of datasets and noise types. We then proposed a method to detect and remove noise from the input without using any extra models, training, or prior information about noise types, and demostrated its efficacy. In future work, we plan to investigate what makes certain models more susceptible to specific noise types. We also plan to carry out experiments with real-world noisy data rather than synthetically created noise.

References

[1] R. Chan, K. Lis, S. Uhlemeyer, H. Blum, S. Honari, R. Siegwart, M. Salzmann, P. Fua, and M. Rottmann. Segmentmeifyoucan: A benchmark for anomaly segmentation. *arXiv preprint arXiv:2104.14812*, 2021.

[2] B. Gliwa, I. Mochol, M. Biesek, and A. Wawer. SAMSum corpus: A human-annotated dialogue dataset for abstractive summarization. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 70–79, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5409. URL `https://aclanthology.org/D19-5409`.

[3] D. Hendrycks, S. Basart, M. Mazeika, M. Mostajabi, J. Steinhardt, and D. Song. Scaling out-of-distribution detection for real-world settings. *arXiv preprint arXiv:1911.11132*, 2019.

[4] H. Husain, H.-H. Wu, T. Gazit, M. Allamanis, and M. Brockschmidt. Codesearchnet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*, 2019.

[5] H. Jing, D. Lopresti, and C. Shih. Summarization of noisy documents: a pilot study. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, pages 25–32, 2003.

[6] V. Karpukhin, O. Levy, J. Eisenstein, and M. Ghazvininejad. Training on synthetic noise improves robustness to natural noise in machine translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 42–47, 2019.

[7] B. Kim, H. Kim, and G. Kim. Abstractive summarization of reddit posts with multi-level memory networks, 2018.

[8] B. Kim, H. Kim, and G. Kim. Abstractive summarization of Reddit posts with multi-level memory networks. 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 2519–2531, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1260. URL `https://www.aclweb.org/anthology/N19-1260`.

[9] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. 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, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL `https://aclanthology.org/2020.acl-main.703`.

[10] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://aclanthology.org/W04-1013`.

[11] A. Meechan-Maddon. The effect of noise in the training of convolutional neural networks for text summarisation, 2019.

[12] J. Mukhoti, J. van Amersfoort, P. H. Torr, and Y. Gal. Deep deterministic uncertainty for semantic segmentation. *arXiv preprint arXiv:2111.00079*, 2021.

[13] S. Narayan, S. Cohen, and M. Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807. Association for Computational Linguistics, 2018.

[14] X. Niu, P. Mathur, G. Dinu, and Y. Al-Onaizan. Evaluating robustness to input perturbations for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8538–8544, 2020.

[15] D. Peskov, J. Barrow, P. Rodriguez, G. Neubig, and J. Boyd-Graber. Mitigating noisy inputs for question answering. *arXiv preprint arXiv:1908.02914*, 2019.

[16] 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. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[17] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[18] J. Ren, J. Luo, Y. Zhao, K. Krishna, M. Saleh, B. Lakshminarayanan, and P. J. Liu. Out-of-distribution detection and selective generation for conditional language models, 2022. URL https://arxiv.org/abs/2209.15558.

[19] 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)*, pages 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://aclanthology.org/P17-1099.

[20] V. Vaibhav, S. Singh, C. Stewart, and G. Neubig. Improving robustness of machine translation with synthetic noise. 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 1916–1920, 2019.

[21] J. Zhang, Y. Zhao, M. Saleh, and P. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

A Appendix

Table A.1: ROUGE scores on clean input and changes when adding different kinds of noise, and after the noise is filtered out using our method (Noise amount: 0.5)

Variant	**Noise type**	**ROUGE-1 / 2 / L**		
		XSum		
		Small	Base	Large
Clean	-	43.35 / 20.49 / 35.73	47.03 / 23.72 / 39.05	48.92 / 25.65 / 40.95
Noisy	Code	31.54 / 12.44 / 25.07	38.74 / 17.34 / 31.43	47.53 / 24.48 / 39.63
	Emoji	31.79 / 15.10 / 26.29	40.08 / 20.32 / 33.24	47.86 / 25.02 / 40.16
	Randomsent	32.38 / 13.10 / 26.09	36.54 / 16.63 / 29.87	42.67 / 21.06 / 35.38
	URL	36.47 / 15.45 / 29.42	37.56 / 16.91 / 30.55	47.37 / 24.45 / 39.64
Filtered	Code	38.72 / 17.00 / 31.61	43.50 / 20.98 / 35.94	48.55 / 25.45 / 40.64
	Emoji	42.94 / 20.24 / 35.38	46.04 / 23.29 / 38.27	48.41 / 25.41 / 40.61
	Randomsent	39.84 / 17.81 / 32.42	43.88 / 21.30 / 36.11	46.65 / 23.84 / 38.85
	URL	41.86 / 19.30 / 34.43	45.69 / 22.69 / 37.80	48.41 / 25.34 / 40.54
		CNN-Dailymail		
		Small	Base	Large
Clean	-	44.50 / 22.74 / 32.27	45.70 / 23.72 / 33.43	46.20 / 24.08 / 33.61
Noisy	Code	36.74 / 16.58 / 26.50	38.54 / 17.22 / 27.32	42.23 / 20.32 / 30.23
	Emoji	43.97 / 22.11 / 31.35	45.25 / 23.21 / 32.77	45.95 / 23.74 / 33.27
	Randomsent	42.63 / 21.02 / 30.16	44.09 / 22.17 / 31.40	44.81 / 22.75 / 32.07
	URL	42.19 / 20.57 / 30.17	43.60 / 21.45 / 31.05	44.89 / 22.69 / 32.27
Filtered	Code	33.64 / 15.60 / 24.33	36.66 / 16.97 / 26.31	43.45 / 21.78 / 31.46
	Emoji	44.14 / 22.27 / 31.68	45.30 / 23.40 / 33.00	45.84 / 23.68 / 33.17
	Randomsent	42.99 / 21.34 / 30.58	44.26 / 22.35 / 31.70	45.16 / 23.08 / 32.51
	URL	42.77 / 21.27 / 30.87	43.89 / 21.97 / 31.65	45.34 / 23.43 / 32.83
		Samsum		
		Small	Base	Large
Clean	-	50.56 / 25.66 / 42.16	51.73 / 27.80 / 43.64	53.50 / 29.53 / 45.68
Noisy	Code	44.81 / 21.32 / 37.62	48.32 / 25.29 / 41.30	50.24 / 26.85 / 43.29
	Emoji	49.27 / 24.41 / 41.54	50.75 / 27.37 / 43.30	53.31 / 29.25 / 45.70
	Randomsent	39.81 / 17.27 / 32.31	42.79 / 21.30 / 35.83	42.22 / 21.35 / 35.85
	URL	46.46 / 22.25 / 38.60	48.31 / 25.22 / 41.21	50.51 / 27.57 / 43.24
Filtered	Code	49.22 / 24.56 / 41.24	50.70 / 26.94 / 43.07	52.43 / 28.87 / 45.23
	Emoji	49.00 / 24.41 / 41.42	50.49 / 27.25 / 43.03	53.32 / 29.21 / 45.64
	Randomsent	47.36 / 23.31 / 39.43	49.47 / 25.64 / 41.60	50.40 / 26.56 / 42.89
	URL	49.65 / 25.16 / 41.58	51.29 / 27.63 / 43.39	52.56 / 28.70 / 45.07
		Reddit-TIFU		
		Small	Base	Large
Clean	-	24.06 / 7.81 / 19.86	26.74 / 9.20 / 21.95	27.45 / 9.65 / 22.56
Noisy	Code	17.95 / 5.74 / 14.87	18.72 / 6.21 / 15.46	20.35 / 6.91 / 16.85
	Emoji	20.25 / 6.47 / 16.65	20.09 / 7.14 / 16.51	22.51 / 7.90 / 18.55
	Randomsent	21.15 / 6.62 / 17.18	22.09 / 7.14 / 18.08	21.47 / 7.09 / 17.73
	URL	21.02 / 6.66 / 17.23	24.25 / 8.09 / 19.76	24.55 / 8.17 / 20.26
Filtered	Code	20.98 / 6.83 / 17.37	22.24 / 7.58 / 18.27	24.31 / 8.46 / 20.15
	Emoji	23.49 / 7.71 / 19.42	21.95 / 7.59 / 17.99	23.79 / 8.40 / 19.59
	Randomsent	23.05 / 7.32 / 18.81	25.57 / 8.64 / 20.78	26.37 / 9.12 / 21.59
	URL	21.96 / 7.10 / 17.94	24.88 / 8.44 / 20.37	25.74 / 8.84 / 21.14

```
w_end = w_offset + wrg. block2 = superints[idx, :, edgearr[idx, 2]:edgearr[idx, 3]]. McLaren executive director Zak Brown
said he regarded the 17-year-old Englishman as "a fabulous prospect". yield Edge(source=self.resources[identifier_source],.
-----. if getextent:. Norris won two Formula Renault 2.0 titles last season and will move to the European Formula Three
series in 2017. self.shutdown_network(). objectinfo['kmag']),. Triple world champion Lewis Hamilton and Red Bull's Max
```

Figure A.1: Sample excerpt from an article from XSum dataset corrupted with code noise.