PARAPHRASE LOSS FOR ABSTRACTIVE SUMMARIZATION

Daniele Rege Cambrin & Paolo Garza
Department of Automatic and Informatics
Politecnico di Torino, Turin, 10129, Italy
{daniele.regecambrin,paolo.garza}@polito.it

ABSTRACT

Fine-tuned models for conditional generation grant state-of-the-art results for abstractive summarization. They achieve high scores by leveraging great amounts of training data and technically "unlimited" training time coupled with the simple cross-entropy loss. We argue that, similarly to the computer vision domain, the natural language processing tasks should be solved using more complex and task-specific losses. These are more robust and improve the results without increasing the training data, applying augmentation approaches, or increasing the overall number of steps. In this work, we propose a new loss term based on paraphrases of the summaries, coupled with cross-entropy, to train models for abstractive summarization, improving state-of-the-art results without increasing the required timesteps.

1 INTRODUCTION

Models for conditional generation proved to be more effective than ones for causal generation in single document abstractive summarization (Syed et al., 2021), being capable of better capturing the style and the content of desired summaries. BART (Lewis et al., 2019) proved to generalize better than other encoder-decoder models (Demeter et al., 2023), being bidirectional and having a more robust pre-training. State-of-the-art generative tasks in NLP are based on cross-entropy loss, posing a single objective in the optimization process, which tends to maximize only the global accuracy. Conversely, in computer vision, particularly for semantic segmentation (Jadon, 2020), many loss functions were designed to improve the results without changing the architectures or the amount of used data. To our knowledge, only for non-generative NLP tasks, the effectiveness of changing the objective function was proven by adapting the dice loss to deal with data-imbalanced tasks (Li et al., 2020). Multiple objective losses (Taghanaki et al., 2019) provide more robust and performing models without changing the amount of training data and the architecture. The only difficulty is to tweak the balance between the different components of the loss. This work focused on single document abstractive summarization using BART and a multiple objective loss composed of cross-entropy and a simple term computed on the paraphrases of the ground truth summaries. In this way, compared to augmentation techniques, we do not need to increase the amount of data and, hence, the number of timesteps needed for training. However, we can include the generalization contribution of the paraphrases in a loss term. For reproducibility, we released our code on an anonymous repository[1].

2 METHODOLOGY

We have a dataset S composed of pairs (text, golden summary). Our goal is to train a single document abstractive summarizer. The proposed approach uses paraphrases of the golden summaries to train a more general and robust model. To create paraphrases without involving human annotators, we leverage the power of modern deep-learning paraphrase models, which have proven effective in text augmentation (Gao et al., 2020). We used an open-sourced paraphrase model based on T5[2]. For

[1] https://github.com/DarthReca/paraphrase-loss
[2] https://huggingface.co/humarin/chatgpt_paraphraser_on_T5_base

each golden summary of the training set, we generate N paraphrases, obtaining a new dataset S_p. More information about the generation can be found in Appendix B.

Given a fixed number of input tokens L and the size of the token vocabulary V, we create a distribution matrix D for each golden summary of size $L \times V$. In each position (x, y) of the matrix, we have the frequency of token y appearing at position x according to the golden summary and the N paraphrases. The procedure can be seen in Figure 1. In this way, we can add a loss term exploiting the KL-divergence that compares the prediction of the summarizer (model) with the distribution matrix D, obtaining the following combo loss:

$$L = \alpha L_{ce}(P, G) + (1 - \alpha) L_{kl}(P, D) \tag{1}$$

where L_{ce} is cross-entropy loss, L_{kl} is the KL-divergence loss, P are the prediction of the summarizer, and G are the golden summary tokens. The intuition is that P should have a distribution similar to both the golden summary and the paraphrases. If we employ data augmentation, we need $N \cdot |S_p|$ iterations to generate the entire augmented dataset, while using our loss, we need only $|S_p|$ iterations.

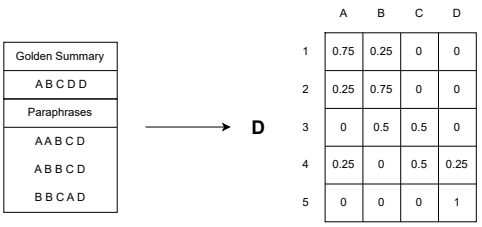

Figure 1: Creation of D matrix

3 Experimental Results

We tested our loss on two well-known benchmark datasets for summarization: CNN-DailyMail (Hermann et al., 2015) and XSum (Narayan et al., 2018). We used the original train-validation-test splits. We evaluate the effects on the BART-base model in an encoder-decoder fashion using ROUGE and BERT-Score. All settings are listed in Appendix A. We run our experiments on an A4000 GPU. In Table 1, we report the metrics across the two datasets, comparing the classical setting with the usage of our proposed combo loss. The improvements are $+0.1\%$ on average on CNN and $+0.02\%$ on average on XSum. To better understand the differences, we compute the Jaccard Index between the generated text using the paraphrase loss and without using it. On CNN, the index is 0.69 ± 0.23, while on XSum is 0.71 ± 0.28. The two generated texts are equal for some samples, while there are many syntactical differences for many others. The meaning is similar but expressed differently. You can see a qualitative difference in Appendix C.

Table 1: Results on the CNN and XSum test sets over 3 runs. *BS* is BERT-Score F1-score, while *RX* are the various ROUGE F1-scores. BART-PL stands for BART with paraphrase loss. The * indicates values for which $p < 0.05$ according to the t-test.

		BS	R1	R2	RL	RL-Sum
CNN	BART	0.8861±0.0015	0.4051±0.0059	0.1888±0.0028	0.2847±0.0017	0.3775±0.0054
	BART-PL	**0.8868±0.0011***	**0.4093±0.0006***	**0.1909±0.0006***	**0.2857±0.0008***	**0.3814±0.0007***
XSum	BART	**0.9110±0.0019**	0.3775±0.0112	0.1614±0.0121	**0.3078±0.0116**	**0.3077±0.0115**
	BART-PL	**0.9110±0.0019**	**0.3784±0.0117***	**0.1619±0.0120**	0.3075±0.0110	0.3076±0.0112

4 Conclusion

In this work, we demonstrated how an extra term of the loss function affects the results of single document abstractive summarization, improving a state-of-the-art model. In future works, we plan to investigate improvements in loss functions on causal generation and more complex tasks in which the benefits could be more evident.

URM STATEMENT

The authors acknowledge that Daniele Rege Cambrin meets the URM criteria of ICLR 2024 Tiny Papers Track.

REFERENCES

David Demeter, Oshin Agarwal, Simon Ben Igeri, Marko Sterbentz, Neil P. Molino, John M. Conroy, and Ani Nenkova. Summarization from leaderboards to practice: Choosing A representation backbone and ensuring robustness. *CoRR*, abs/2306.10555, 2023. doi: 10.48550/ARXIV.2306.10555. URL https://doi.org/10.48550/arXiv.2306.10555.

Silin Gao, Yichi Zhang, Zhijian Ou, and Zhou Yu. Paraphrase augmented task-oriented dialog generation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 639–649, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.60. URL https://aclanthology.org/2020.acl-main.60.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

Shruti Jadon. A survey of loss functions for semantic segmentation. In *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pp. 1–7, 2020. doi: 10.1109/CIBCB48159.2020.9277638.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019. URL http://arxiv.org/abs/1910.13461.

Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. Dice loss for data-imbalanced NLP tasks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 465–476, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.45. URL https://aclanthology.org/2020.acl-main.45.

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

Ayesha Ayub Syed, Ford Lumban Gaol, and Tokuro Matsuo. A survey of the state-of-the-art models in neural abstractive text summarization. *IEEE Access*, 9:13248–13265, 2021. doi: 10.1109/ACCESS.2021.3052783.

Saeid Asgari Taghanaki, Yefeng Zheng, S Kevin Zhou, Bogdan Georgescu, Puneet Sharma, Daguang Xu, Dorin Comaniciu, and Ghassan Hamarneh. Combo loss: Handling input and output imbalance in multi-organ segmentation. *Computerized Medical Imaging and Graphics*, 75: 24–33, 2019.

A TRAINING AND TESTING SETTINGS

Table 2 reports the training settings employed. If not specified, the parameter is left to its default value according to the employed library. More details in the repository.

B PARAPHRASE GENERATION SETTINGS

We employed the default generation settings suggested in the model to create the paraphrases. In this way, we got $N = 5$. We released the two generated train sets on Zenodo[3]. We do not generate

[3] https://zenodo.org/records/10254976

Table 2: Training and Testing settings

Batch Size	8	Learning rate	5e-5
Learning Rate Schedule	Linear	Epochs	1
Warmup Steps	500	Seed	2, 42, 142
Optimizer	Adam	Max Generation Length	XSum: 64, CNN: 128
		α	0.1

any paraphrases for the validation and test sets. The mean required amount of time to generate all the paraphrases for a sample is $\approx 26s$ on Intel(R) Xeon(R) CPU X5650 @ 2.67GHz, and the mean GFLOPs required are ≈ 10.8.

C EXAMPLES

In Figure 2, we report qualitative examples on the XSum dataset.

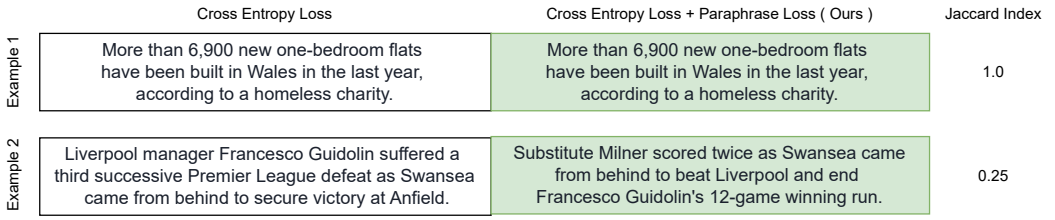

Figure 2: Comparison between generated summaries on XSum with and without paraphrase loss.

D LEXICAL ANALYSIS

We have analyzed the lexical differences between the predictions generated using paraphrase loss and standard cross-entropy. We have created a dictionary for each model for each of the aforementioned seeds after lemmatization and stopword removal, tracking the number of appearances of each word. The two dictionaries show $p < 0.001$ according to the t-test. Around 26% of the words are used by one model, while the other version does not use them. Paraphrase loss models employ ≈ 100 more new words than standard training, providing a more varied dictionary. Ranking the words in the dictionary based on their count, $\approx 99.99\%$ of the ranks are different.

E GENERALIZATION ANALYSIS

In this section, we report the results of transfer learning on the two datasets to test the generalization of the trained models. From Table 3, we can see the performance of models trained on XSum tested on CNN and vice versa. We get average improvements of $+0.04\%$ on Xsum and $+0.05\%$ on CNN. This suggests the additional component in the loss function contributes to better generalizing the predictions.

Table 3: Results on the CNN and XSum test sets over 3 runs using transfer learning. *BS* is BERT-Score F1-score, while *RX* are the various ROUGE F1-scores. BART-PL stands for BART with paraphrase loss. The * indicates values for which $p < 0.05$ according to the t-test.

		BS	R1	R2	RL	RL-Sum
CNN	BART	0.8613±0.0002	0.2149±0.0025	0.0596±0.0004	0.1519±0.0010	0.1895±0.0018
	BART-PL	**0.8614±0.0002**	**0.2159±0.0014***	**0.0597±0.0006**	**0.1523±0.0007**	**0.1905±0.0009***
XSum	BART	0.8548±0.0001	0.1922±0.0002	**0.0273±0.0001**	0.1278±0.0003	0.1547±0.0009
	BART-PL	**0.8553±0.0003***	**0.1927±0.0000***	**0.0273±0.0001**	**0.1289±0.0007***	**0.1548±0.000**