Compare Encoder-Decoder, Encoder-Only, and Decoder-Only Architectures for Text Generation on Low-Resource Datasets

Pei-Xuan Cai[1], Yao-Chung Fan[1], and Fang-Yie Leu[2(✉)]

[1] National Chung Hsing University, Taichung, Taiwan
[2] TungHai University, Taichung, Taiwan
leufy@thu.edu.tw

Abstract. Natural language generation (NLG) tasks have received significant research attention in recent years. For tackling various NLG tasks, the Transformer [27] is now consensus to be employed as a fundamental building block. In the literature, there are three main Transformer variants for NLG: full Transformer, Encoder-Only (only using the encoder part of the Transformer), and Decoder-Only (only using the decoder part). A natural question to ask is: which architecture is the best choice. According to previous studies, when the amount of training dataset is sufficient, using the full Transformer is the priority choice for NLG tasks. However, for the insufficient training dataset setting, we find this is not the case. In this paper, we report experiment results of applying the three architectures to four different tasks under low-resource settings. In contrast to the conclusion by previous study, we find that there are no consistent results indicating which architecture is the best under low-resource dataset settings. Further, based on the experiment results, we comment on the architecture selection under the low-resource dataset consideration.

1 Introduction

Natural language generation (NLG) tasks based on deep learning techniques have received significant research attention in recent years. The applications of natural language generation are text summarization [17,31] (generating summarization of a given article), machine translation [8,24] (generating text in a different language based on the source text), and question generation task [4,29] (automatically generating questions based on a given context paragraph). Such NLG applications play important roles for AI applications nowadays.

For NLG, a common practice is to employ the Transformer [27]. The Transformer is an encoder-decoder architecture. The encoder contains encoding layers processing the input iteratively one layer after another, while the decoder contains decoding layers for text generation. In the literature, there are three variants of using the Transformer for language generation, which we review them as follows.

L. Barolli (Ed.): BWCCA 2021, LNNS 346, pp. 216–225, 2022.
https://doi.org/10.1007/978-3-030-90072-4_23

- Encoder-Decoder (full Transformer): A direct employment of the Transformer for text generation. In this architecture, input is encoded by the encoder and the decoder conducts the shifted right [27] operation to predict the next token. Representatives for such an architecture are BERT2BERT model [24] and BART model [14].
- Encoder-Only: In this line of work, only the encoder part of the Transformer is used for text generation. Representatives for such a architecture are BERT-HLSQG model [4] and BERTGEN model [19]. The main idea is to predict the next token by iterating over the encoder.
- Decoder-Only: In this line of work, only the decoder part of the Transformer is used. A well-known representative is GPT-2 model [22]. The idea is to predict the next token by continuously iterating the Decoder.

Although all three architectures have the own advocators, by our investigation, the state-of-the-art (SOTA) result of various NLG tasks are based on the full Transformer architecture. Therefore, the consensus is that when training dataset is sufficient, the full transformer architecture would be the best choice.

However, their comparison (the three transformer variant architectures) under low-resource datasets settings remained under-explored. Example scenario for low-resource text generation setting are matching of medical questions [18], reading comprehension of Persian [12], or machine translation from Cherokee to English [32].

In this paper, we investigate the performance difference of the full Transformer, Encoder-Only, and Decoder-Only architecture under low-resource dataset setting. Specifically, in this paper, we mainly explore the research question:

Which architecture will be the best choice for text generation task under low-resource dataset setting?

Specifically, this paper reports experiment results of applying the three architectures to four different tasks (Paraphrase Generation, Machine Translation, Question Generation, and Abstractive Text Summarization). In contrast to the conclusion drawn by rich dataset settings, we find that there are no consistent results indicating which architecture is the best under low-resource dataset settings.

By experiment observation, we find the following observations for text generation under low-resource datasets settings.

- First, NLG tasks requiring semantic understanding, such as abstractive text summarization and question generation, are better to tackle by using the full Transformer.
- Second, the Encoder-Only architecture shows better performance for a NLG task requiring only the lexical reformation or rewriting, such as paraphrase generation.
- Third, the Decoder-Only architecture seems not a good choice when a low-resource dataset setting is considered.

2 Related Work

In the literature, there are three main Transformer variants for NLG: full Transformer, Encoder-Only, and Decoder-Only architectures. The full Transformer's

Table 1. The current SOTA research on rich-resource generation tasks.

Task	Dataset	Architecture	SOTA models
Paraphrase Generation	Quora Question Pairs	Encoder-Decoder	[11]
Machine Translation	WMT14 German-English	Encoder-Decoder	[13]
Question Generation	SQuAD 1.1	Encoder-Decoder	[29]
Abstractive Text Summarization	CNN/Daily Mail	Encoder-Decoder	[15]

representative is the BERT2BERT model [24], the Encoder-Only's representative is the BERT-GEN model [19], and the Decoder-Only's representative is the GPT-2 model [22]. Three architectures have been employed for various NLG applications. For evaluating the performance of NLG models, there are four commonly used benchmark tasks: Paraphrase Generation, Machine Translation, Question Generation, and Abstractive Text Summarization. We find that although the models have the own advocators and the SOTA results of the four tasks are mainly the full Transformer. Please refer to Table 1, from which one can see the SOTA performance of the tasks are the full Transformer. In fact, the general consensus of selecting NLG architecture is to use the full Transformer.

The works [1,24] compare the performance of the full Transformer and Decoder-Only architecture. The study [24] is to replace the weights of full Transformer with pre-trained checkpoints and compare them with GPT-2. [1] is to pre-train the full transformer with the information of the relevant task and compare it with GPT-2. The conclusions made by the two studies indicate that the full Transformer is the winner.

However, we would like to note that the existing comparison are based on the rich training setting. To our best knowledge, the comparison under low-resource setting is not explored. In this paper, we use the datasets listed in Table 1 to compare the three architectures under low-resource settings.

With regard to low-resource settings, research investigated by [9] points out that the amount of so-called low-resource varies for different tasks. For example, the work in [30] treats 350K as a low-resource for question response generation tasks. Yet another example is that [6] treats 10K as low-resource on abstractive text summarization tasks. In order to maintain uniformity and to consider a more demanding resource situation, we set to take 1K and 3K of the training data for each task to compare the models.

Note that there are many techniques for addressing low-resource setting, such as data augmentation [5] or transfer learning [26] for making effective use of low-resources datasets. We would like to note that the goal of this study to compare the strengths and weaknesses of the architectures directly trained with the given insufficient data.

3 Performance Comparison

In this section, we conduct experiments on the mentioned four generation tasks to observe the performance difference of the compared architectures.

For a fair comparison, the models are all trained with initial parameters whose weights are randomly set. Furthermore, we consider two low-resource dataset setting: 3k and 1k settings; we randomly select 3000 and 1000 instances from the original datasets as training datasets for simulating the low-resource dataset setting. We use the original released testing dataset setting for performance evaluation. The scores are the average of the three random selected training sets. We evaluate the performance through the evaluation package released by [25]. The package includes BLEU 1, BLEU 2, BLEU 3, BLEU 4 [21], METEOR [2] and ROUGE [16] evaluation scripts.

3.1 Model Setup

We built the Encoder-Decoder, Encoder-Only, and Decoder-Only architectures based on the PyTorch version of BERT [1], and initialized all weights randomly for training each task. All tasks use the BERT-Base Cased vocabulary (28996 words) and follow the [27] settings, with the hidden dimension set to 512, attention heads set to 8, and a feed-forward layer set to 2048.

Note that we adjust the layers of the compared models to have a fair comparison. This is because if the architectures are using the same number of layers, there will be significant difference of the total number of parameters for the architecture, bringing the concern of unfair comparison. Therefore, we adjust the number of layers of the implemented architectures to enable a match/close parameter numbers. We set the Encoder-Decoder to have one layer only (an encoder layer and a decoder layer). On the other hands, the number of layers of Encoder-Only and Decoder-Only is set to 7. The total number of parameters of each implemented model is near 52M.

The dropout probability between transformer layers was set to 0.1. The Adamax optimizer is applied during training with an initial learning rate of 5e-5. The batch size for the update is set at 50. The Epoch is set to 60 for Encoder-Only and 100 for Decoder-Only and Encoder-Decoder. All of our models are trained by using two TITAN RTX GPUs.

3.2 Paraphrase Generation

Paraphrase Generation is a task that take a source sentence to generate a sentence with different syntax structure but the same semantic meaning. We use GLUE-QQP dataset [28] to compare the model performance.

GLUE-QQP: A collection of question pairs collected from the community question-answering website Quora are tagged with either 0 or 1, with 1 meaning the two sentences are semantically identical, and 0 the other way around. In the dataset, there are 134,378 instances labeled as 1. We set the maximum length of a source sentence to 105 and a target sentence to 97.

Results. Table 2 shows GLUE-QQP validation results. We see Encoder-Only show the best performing results on both 1K and 3K. We think this is related to

[1] https://github.com/huggingface/transformers.

the characteristic of the paraphrase generation task. Since paraphrase generation is mainly about swapping or grammatical changes to the words, Encoder-Only learns one token at a time, so it can effectively catch the points that need to be changed.

3.3 Machine Translation

Machine Translation task, which translates the source text of one language into the text of another language. We conduct our comparisons on WMT14 German-English Newstest2014 [3].

WMT14 German-English Newstest 2014: There are 4.5M sentence pairs in the original datasets. We set the goal to translate German to English. We set the maximum length of German sentence to 234 and English sentence to 124.

Results. Table 3 shows WMT14 German-English Newstest 2014 test results. We see Encoder-Decoder obtains the best scores on both BLEU 3 and BLEU 4. We think that machine translation tasks requires deep understanding of grammatical differences between different languages. Under this task characteristic, the full Transformer is a better fit, where the encoder takes charge of understanding the source language and the decoder responses for target language generation.

Table 2. GLUE-QQP evaluation results

Model	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE-L
1K						
Encoder-Decoder	32.17	17.65	11.23	7.25	12.76	33.18
Encoder-Only	**33.00**	**18.31**	**11.85**	**8.10**	**13.29**	**33.36**
Decoder-Only	25.45	13.94	8.83	5.72	12.18	30.50
3K						
Encoder-Decoder	39.18	24.43	16.90	12.18	17.08	39.34
Encoder-Only	**44.70**	**28.74**	**19.96**	**14.47**	**20.81**	**44.82**
Decoder-Only	41.55	26.94	18.79	13.56	19.81	43.01

Table 3. WMT14 German-English Newstest 2014 test results

Model	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE-L
1K						
Encoder-Decoder	19.27	**5.13**	**1.26**	**0.40**	5.51	**17.03**
Encoder-Only	**19.31**	4.63	0.78	0.19	**5.60**	16.18
Decoder-Only	18.81	4.84	1.18	0.37	5.51	16.67
3K						
Encoder-Decoder	20.45	5.85	**1.80**	**0.66**	6.64	18.01
Encoder-Only	18.67	5.02	1.20	0.34	6.24	16.51
Decoder-Only	**21.32**	**6.05**	1.76	0.63	**6.69**	**18.12**

3.4 Question Generation

Question generation task, which takes a context text and an answer phase as input and generates a question corresponding to the given answer phase. We evaluate the performance on SQuAD 73K [7]. The SQuAD contains 536 articles with 100K questions (and the corresponding answers) about these articles.

SQuAD 73K: Based on the setting by [7], SQuAD 73k [23] is divided into the training data with a training set (80%), a development set (10%) and a test set (10%). We set the maximum length of the context to 422, question to 50 and answer to 15.

Results. Table 4 shows SQuAD 73K test results. The best results were obtained by Encoder-Decoder in both 1k and 3k experiment setting. We think that the question generation task needs to understand the context and the answer before the relevant questions can be generated. A conclusion similar to the language translation is that for understanding paragraph level information, the Encoder-Decoder is the most suitable.

Table 4. SQuAD 73K test results

Model	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE-L
1K						
Encoder-Decoder	**24.18**	**8.66**	**4.34**	**2.42**	**6.89**	**25.71**
Encoder-Only	16.21	3.78	0.99	0.19	5.52	20.67
Decoder-Only	21.15	6.81	3.15	1.45	6.30	23.50
3K						
Encoder-Decoder	**24.71**	**9.25**	**4.78**	**2.78**	**7.01**	**25.55**
Encoder-Only	11.26	3.19	1.11	0.42	5.43	15.18
Decoder-Only	19.61	6.93	3.31	1.57	6.15	22.49

Table 5. CNN/DailyMail test results

Model	BLEU 1	BLEU 2	BLEU 3	BLEU 4	METEOR	ROUGE-L
1K						
Encoder-Decoder	12.64	**3.67**	**1.03**	**0.42**	3.83	**12.11**
Encoder-Only	**13.98**	2.25	0.21	0.03	**4.18**	11.18
Decoder-Only	2.53	0.64	0.17	0.06	2.42	8.75
3K						
Encoder-Decoder	**15.50**	**4.46**	**1.31**	**0.51**	**5.21**	**13.37**
Encoder-Only	4.94	0.64	0.1	0.03	2.66	7.84
Decoder-Only	2.04	0.52	0.18	0.08	2.04	7.73

3.5 Abstractive Text Summarization

Abstractive Text Summarization task, whose goal is to take an article to generate a coherent and semantically correct abstract. We evaluate our model performance on CNN/DailyMail [10,20].

CNN/DailyMail: The corpus has 286,817 training pairs, 13,368 validation pairs and 11,487 test pairs. We set the maximum length of the article to 435 and corresponded abstract to 73.

Results. Table 5 shows CNN/DailyMail test results. Encoder-Decoder still gets the best scores on both 1K and 3K ROUGE-L. We think that the abstractive text summarization task still requires understanding the context of the article in order to generate a relevant summary. Therefore, the Encoder-Decoder architecture is again a best fit for the abstractive text summarization task.

3.6 Result Discussion

Based on the experimental results, we think that the full Transformer can efficiently leverage the bidirectional information captured by the encoder component and leverage the auto-regressive capability of the decoder component for coherent text generation, which is suitable for text generation tasks requiring semantic understanding. On the other hand, Encoder-Only can generate the next word that may appear by using bidirectional information, but it cannot be utilized and generated efficiently. However, it provides excellent performance in lexical reformation or rewriting tasks, such as paraphrase generation. Decoder-Only can only use the previous information to generate the next word that may appear, but it cannot use the previous information to do the action of mutual consideration, so it is inferior to the full Transformer in tasks that require paragraph level semantic understanding.

4 Conclusion

In this paper, we conduct experiments to compare three main NLG architectures to see which one is more effective for each task under low-resource setting scenario. Different to the previous conclusion (the full Transformer always a winner) on rich dataset setting, we find Encoder-Only architecture will be good for tasks requiring only text reformation or rewriting, such as paraphrase generation. However, if a NLG task requires understanding paragraph level semantic, the full Transformer is still the best choice.

Acknowledgments. This work is partially supported by MOST 110-2218-E-005-008-MBK, TWISC project, Taiwan.

References

1. Ahmad, W.U., Chakraborty, S., Ray, B., Chang, K.: Unified pre-training for program understanding and generation. In: Toutanova, K., et al. (eds.) Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, 6–11 June 2021, pp. 2655–2668. Association for Computational Linguistics (2021)
2. Banerjee, S., Lavie, A.: METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In: Goldstein, J., Lavie, A., Lin, C., Voss, C.R. (eds.) Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, 29 June 2005, pp. 65–72. Association for Computational Linguistics (2005)
3. Bojar, O., et al.: Findings of the 2014 workshop on statistical machine translation. In: Proceedings of the Ninth Workshop on Statistical Machine Translation, WMT@ACL 2014, Baltimore, Maryland, USA, 26–27 June 2014, pp. 12–58. The Association for Computer Linguistics (2014)
4. Chan, Y., Fan, Y.: A recurrent BERT-based model for question generation. In: Fisch, A., Talmor, A., Jia, R., Seo, M., Choi, E., Chen, D. (eds.) Proceedings of the 2nd Workshop on Machine Reading for Question Answering, MRQA@EMNLP 2019, Hong Kong, China, 4 November 2019, pp. 154–162 (2019)
5. Dehouck, M., Gómez-Rodríguez, C.: Data augmentation via subtree swapping for dependency parsing of low-resource languages. In: Scott, D., Bel, N., Zong, C. (eds.) Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), 8–13 December 2020, pp. 3818–3830. International Committee on Computational Linguistics (2020)
6. Dong, L., et al.: Unified language model pre-training for natural language understanding and generation. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, Vancouver, BC, Canada, 8–14 December 2019, pp. 13042–13054 (2019)
7. Du, X., Shao, J., Cardie, C.: Learning to ask: neural question generation for reading comprehension. In: Barzilay, R., Kan, M. (eds.) Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, 30 July–4 August, Long Papers, vol. 1, pp. 1342–1352. Association for Computational Linguistics (2017)
8. Guo, J., Zhang, Z., Xu, L., Wei, H., Chen, B., Chen, E.: Incorporating BERT into parallel sequence decoding with adapters. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, 6–12 December 2020, virtual (2020)
9. Hedderich, M.A., Lange, L., Adel, H., Strötgen, J., Klakow, D.: A survey on recent approaches for natural language processing in low-resource scenarios. In: Toutanova, K., et al.(eds.) Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, 6–11 June 2021, pp. 2545–2568. Association for Computational Linguistics (2021)
10. Hermann, K.M., et al.: Teaching machines to read and comprehend. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural

Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, Montreal, Quebec, Canada, 7–12 December 2015, pp. 1693–1701 (2015)
11. Hosking, T., Lapata, M.: Factorising meaning and form for intent-preserving paraphrasing. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, 1–6 August 2021, pp. 1405–1418. Association for Computational Linguistics (2021)
12. Khashabi, D., et al.: ParsiNLU: a suite of language understanding challenges for Persian. CoRR abs/2012.06154 (2020)
13. Kong, X., Zhang, Z., Hovy, E.H.: Incorporating a local translation mechanism into non-autoregressive translation. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, 16–20 November 2020, pp. 1067–1073. Association for Computational Linguistics (2020)
14. Lewis, M., et al.: BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J.R. (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, 5–10 July 2020, pp. 7871–7880. Association for Computational Linguistics (2020)
15. Liang, X., et al.: R-drop: regularized dropout for neural networks. CoRR abs/2106.14448 (2021)
16. Lin, C.Y.: Rouge: a package for automatic evaluation of summaries. Text Summarization Branches Out (2004)
17. Liu, Y., Lapata, M.: Text summarization with pretrained encoders. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, 3–7 November 2019, pp. 3728–3738. Association for Computational Linguistics (2019)
18. McCreery, C.H., Katariya, N., Kannan, A., Chablani, M., Amatriain, X.: Effective transfer learning for identifying similar questions: matching user questions to COVID-19 FAQs. In: Gupta, R., Liu, Y., Tang, J., Prakash, B.A. (eds.) KDD 2020: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, 23–27 August 2020, pp. 3458–3465. ACM (2020)
19. Mitzalis, F., Caglayan, O., Madhyastha, P., Specia, L.: BERTGen: multi-task generation through BERT. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, 1–6 August 2021, pp. 6440–6455. Association for Computational Linguistics (2021)
20. Nallapati, R., Zhou, B., dos Santos, C.N., Gülçehre, Ç., Xiang, B.: Abstractive text summarization using sequence-to-sequence RNNs and beyond. In: Goldberg, Y., Riezler, S. (eds.) Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, 11–12 August 2016, pp. 280–290. ACL (2016)
21. Papineni, K., Roukos, S., Ward, T., Zhu, W.: BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 6–12 July 2002, Philadelphia, PA, USA, pp. 311–318. ACL (2002)

22. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI Blog **1**(8), 9 (2019)
23. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: Squad: 100, 000+ questions for machine comprehension of text. In: Su, J., Carreras, X., Duh, K. (eds.) Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, 1–4 November 2016, pp. 2383–2392. The Association for Computational Linguistics (2016)
24. Rothe, S., Narayan, S., Severyn, A.: Leveraging pre-trained checkpoints for sequence generation tasks. Trans. Assoc. Comput. Linguist. **8**, 264–280 (2020)
25. Sharma, S., Asri, L.E., Schulz, H., Zumer, J.: Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. CoRR abs/1706.09799 (2017)
26. Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C.: A survey on deep transfer learning. In: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, I. (eds.) ICANN 2018. LNCS, vol. 11141, pp. 270–279. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01424-7_27
27. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4–9 December 2017, Long Beach, CA, USA, pp. 5998–6008 (2017)
28. Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., Bowman, S.R.: GLUE: a multi-task benchmark and analysis platform for natural language understanding. In: Linzen, T., Chrupala, G., Alishahi, A. (eds.) Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, 1 November 2018, pp. 353–355. Association for Computational Linguistics (2018)
29. Xiao, D., et al.: ERNIE-GEN: an enhanced multi-flow pre-training and fine-tuning framework for natural language generation. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020, pp. 3997–4003. ijcai.org (2020)
30. Yang, Z., Wu, W., Yang, J., Xu, C., Li, Z.: Low-resource response generation with template prior. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, 3–7 November 2019, pp. 1886–1897. Association for Computational Linguistics (2019)
31. Zhang, H., Cai, J., Xu, J., Wang, J.: Pretraining-based natural language generation for text summarization. In: Bansal, M., Villavicencio, A. (eds.) Proceedings of the 23rd Conference on Computational Natural Language Learning, CoNLL 2019, Hong Kong, China, 3–4 November 2019, pp. 789–797. Association for Computational Linguistics (2019)
32. Zhang, S., Frey, B., Bansal, M.: ChrEn: Cherokee-English machine translation for endangered language revitalization. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, 16–20 November 2020, pp. 577–595. Association for Computational Linguistics (2020)