TextBox: A Unified, Modularized, and Extensible Framework for Text Generation

Junyi Li[1,3]†, **Tianyi Tang**[1]†, **Gaole He**[2], **Jinhao Jiang**[1], **Xiaoxuan Hu**[2], **Puzhao Xie**[2], **Zhipeng Chen**[2], **Zhuohao Yu**[2], **Wayne Xin Zhao**[1,3,4*] and **Ji-Rong Wen**[1,2,3]

[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]School of Information, Renmin University of China
[3]Beijing Key Laboratory of Big Data Management and Analysis Methods
[4]Beijing Academy of Artificial Intelligence, Beijing, 100084, China

`{lijunyi,steven_tang}@ruc.edu.cn` `batmanfly@gmail.com`

Abstract

In this paper, we release an open-source library, called `TextBox`, to provide a unified, modularized, and extensible text generation framework. TextBox aims to support a broad set of text generation tasks and models. In our library, we implement 21 text generation models on 9 benchmark datasets, covering the categories of VAE, GAN, and pretrained language models. Meanwhile, our library maintains sufficient modularity and extensibility by properly decomposing the model architecture, inference, and learning process into highly reusable modules, which allows users to easily incorporate new models into our framework. The above features make TextBox especially suitable for researchers and practitioners to quickly reproduce baseline models and develop new models. TextBox is implemented based on PyTorch, and released under Apache License 2.0 at the link `https://github.com/RUCAIBox/TextBox`.

1 Introduction

Text generation, which has emerged as an important branch of natural language processing (NLP), is often formally referred as natural language generation (NLG) (Li et al., 2021b). It aims to produce plausible and understandable text in human language from input data (*e.g.,* a sequence, keywords) or machine representation. Because of incredible performance of deep learning models, many classic text generation tasks have achieved rapid progress, such as machine translation (Vaswani et al., 2017), dialogue systems (Li et al., 2016b), text summarization (See et al., 2017), graph-to-text generation (Li et al., 2021a), and more.

To facilitate the development of text generation models, a few remarkable open-source libraries have been developed (Britz et al., 2017; Klein et al., 2017b; Miller et al., 2017b; Zhu et al., 2018; Hu et al., 2019). These frameworks are mainly designed for some or a small number of specific tasks, particularly machine translation and dialogue systems. They usually focus on a special kind of techniques for text generation such as generative adversarial networks (GAN), or have limitations in covering commonly-used baseline implementations. Even for an experienced researcher, it is difficult and time-consuming to implement all compared baselines under a unified framework. Therefore, it is highly desirable to re-consider the implementation of text generation algorithms in a unified and modularized framework.

In order to alleviate the above issues, we initiate a project to provide a unified framework for text generation algorithms. We implement an open-source text generation library, called `TextBox`, aiming to enhance the reproducibility of existing text generation models, standardize the implementation and evaluation protocol of text generation algorithms, and ease the development process of new algorithms. Our work is also useful to support several real-world applications in the field of text generation. We have extensively surveyed related text generation libraries and broadly fused their merits into TextBox. The key features and capabilities of our library are summarized in the following three aspects:

• Unified and modularized framework. TextBox is built upon PyTorch (Paszke et al., 2019), which is one of the most popular deep learning frameworks (especially in the research community). Moreover, it is designed to be highly modularized, by decoupling text generation models into a set of highly reusable modules, including data module, model module, evaluation module, and many common components and functionalities. In our library, it is convenient to compare different text generation

† Equal contribution.
* Corresponding author.

Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, pages 30–39, August 1st - August 6th, 2021.

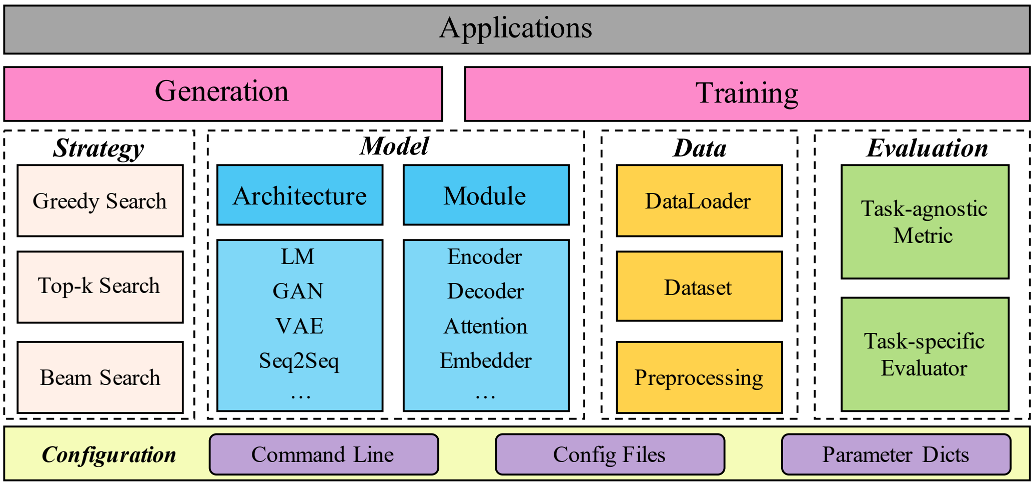

Figure 1: The illustration of the main functionalities and modules in our library TextBox.

algorithms with built-in evaluation protocols via simple yet flexible configurations, or develop new text generation models at a highly conceptual level by plugging in or swapping out modules.

• Comprehensive models, benchmark datasets and standardized evaluations. TextBox contains a wide range of text generation models, covering the categories of variational auto-encoder (VAE), generative adversarial networks (GAN), recurrent neural network (RNN) and pretrained language models (PLMs). We provide flexible supporting mechanisms via the configuration file or command line to run, compare and test these traditional and state-of-the-art algorithms. Based on these models, we implement two major text generation tasks, namely unconditional text generation tasks and conditional text generation tasks (*e.g.,* text summarization and machine translation). To construct a reusable benchmark, we incorporate 9 widely-used datasets with regards to different text generation tasks for evaluation. Our library supports a series of frequently adopted evaluation protocols for testing and comparing text generation algorithms, such as perplexity, BLEU, ROUGE, and Distinct.

• Extensible and flexible framework. TextBox provides convenient interfaces of various common functions or modules in text generation models, *e.g.,* RNN-based and Transformer-based encoders and decoders, pretrained language models, and attention mechanisms. Within our library, users are convenient to choose different API interfaces for building and evaluating their own models. Besides, the interfaces of our library are fully compatible with the PyTorch interface which allows seamless integration of user-customized modules and functions as needed.

2 Architecture and Design

Figure 1 presents the illustration of the main functionalities and modules in our library TextBox. The configuration module at the bottom helps users set up the experimental environment (*e.g.,* hyper-parameters and running details). Built upon the configuration module, the data, model, and evaluation modules form the core elements of our library. In the following, we describe the detailed structure of these three modules.

2.1 Data Module

A major design principle of our library is to support different text generation tasks. For this purpose, data module is the fundamental part to provide various data structures and functions adapting to different generation tasks.

For extensibility and reusability, our data module designs a unified data flow feeding input text into the models. The data flow can be described as: input text → `Dataset` → `DataLoader` → models. The class `Dataset` involves two special data structures, *i.e.,* single sequence and paired sequence, which are oriented to unconditional and conditional text generation tasks, respectively. The single sequence structure requires users to preprocess input text into one sequence per line in input files, while the paired sequence structure requires users to separate the source and target into two files with one sequence per line in each file. Specifically, for conditional text generation, TextBox supports several source formats corresponding to different tasks, *e.g.,* discrete attributes or tokens for attribute-to-text and keyword-to-text generation, a text sequence for machine translation or text summarization, and multiple text sequences for multi-turn dialogue systems. Furthermore, users can also provide additional information as inputs, *e.g.,* background text for agents in dialogues. The implementation of `Dataset` contains many common data preprocessing functionalities, such as converting text into lowercase, word tokenization, and building vocabulary. And the class `Dataloader` is based on the above two data structures, which is responsible for organizing the data stream.

In order to compare different generation models, we have collected 9 commonly-used benchmarks for text generation tasks, which makes it quite convenient for users to start with our library.

2.2 Model Module

To support a variety of models, we set up the model module by decoupling the algorithm implementation from other components and abstracting a set of widely-used modules, *e.g.,* `encoder` and `decoder`. These modules can be flexibly combined following the required interface and then connected with data and evaluation modules. Based on this abstract design, it is convenient to switch between different text generation tasks, and change from one modeling paradigm to another by simply plugging in or swapping out modules.

In addition to modularized design, our library also includes a large number of text generation baseline models for reproducibility. At the current released version, we have implemented 21 baseline models within four main categories of text generation models, namely VAE-based, GAN-based, pretrained language models, and sequence-to-sequence, corresponding to different generation architectures and tasks. For example, GAN-based models consist of `generator` and `discriminator`, and VAE-based models contain `encoder` and `decoder`. We summarize all the implemented models in Table 1. For all the implemented models, we test their performance for unconditional and conditional generation tasks on corresponding benchmarks, and invite a code reviewer to examine the correctness of the implementation. Overall, the extensible and comprehensive model modules can be beneficial for fast exploration of new algorithms for a specific task, and convenient comparison between different models.

In specific, for each model, we utilize two interface functions, *i.e.,* `forward` and `generate`, for training and testing, respectively. These functions are general to various text generation algorithms, so that we can implement various algorithms in a highly unified way. Such a design also enables quick development of new models.

Category	Models	Reference
VAE	LSTM-VAE	(Bowman et al., 2016)
	CNN-VAE	(Yang et al., 2017)
	Hybrid-VAE	(Semeniuta et al., 2017)
	CVAE	(Li et al., 2018)
GAN	SeqGAN	(Yu et al., 2017)
	TextGAN	(Zhang et al., 2017)
	RankGAN	(Lin et al., 2017)
	MaliGAN	(Che et al., 2017)
	LeakGAN	(Guo et al., 2018)
	MaskGAN	(Fedus et al., 2018)
Pretrained Language Model	GPT-2	(Radford et al., 2019)
	XLNet	(Yang et al., 2019)
	BERT2BERT	(Rothe et al., 2020)
	BART	(Lewis et al., 2020)
	ProphetNet	(Qi et al., 2020)
	T5	(Raffel et al., 2020)
Seq2Seq	RNN	(Sutskever et al., 2014)
	Transformer	(Vaswani et al., 2017)
	Context2Seq	(Tang et al., 2016)
	Attr2Seq	(Dong et al., 2017)
	HRED	(Serban et al., 2016)

Table 1: Implemented models in our library TextBox.

In order to improve the quality of generation results, we also implement a series of generation strategies when generating text, such as greedy search, top-k search and beam search. Users are allowed to switch between different generation strategies leading to better performance through setting a hyper-parameter, *i.e.,* `decoding_strategy`. Besides, we add the functions of model saving and loading to store and reuse the learned models, respectively. In the training process, one can print and monitor the change of the loss value and apply training tricks such as warm-up and early-stopping. These tiny tricks largely improve the usage experiences with our library.

2.3 Evaluation Module

It is important that different models should be compared under the unified evaluate protocols, which is useful to standardize the evaluation of text generation. To achieve this goal, we set up the evaluation module to implement commonly-used evaluation protocols for text generation models.

Our library supports both logit-based and word-based evaluation metrics. The logit-based metrics include perplexity (PPL) (Brown et al., 1992) and negative log-likelihood (NLL) (Huszar, 2015), measuring how well the probability distribution or a probability model predicts a sample compared with the ground-truth. The word-based metrics include the most widely-used generation metrics for evaluating lexical similarity, semantic equivalence and diversity. For example, BLEU-n (Papineni et al., 2002) and ROUGE-n (Lin, 2004) measure the ratios of the overlapping n-grams between the generated and real samples, METEOR (Banerjee and Lavie, 2005) measures the word-to-word matches based on WordNet, CIDEr (Vedantam et al., 2015) computes the TF-IDF weights for each n-gram in generated/real samples and CHRF++ (Popovic, 2015) computes F-score averaged on both character- and word-level n-grams. To evaluate the semantic equivalence between generated and real samples, we include BERTScore (Zhang et al., 2020), a metric based on the similarity of sentence embeddings relied on pretrained language model BERT (Devlin et al., 2019). Moreover, Distinct-n and Unique-n (Li et al., 2016a) measures the degree of diversity of generated text by calculating the number of distinct unigrams and bigrams in generated text. Besides, to evaluate the diversity of unconditionally generated samples, we also take into account the Self-BLEU (Zhu et al., 2018) metric. In summary, users can choose different evaluation protocols towards a specific generation task by setting the hyper-parameter, *i.e.,* `metrics`.

In practice, as the model may generate many text pieces, evaluation efficiency is an important concern. Hence, we integrate efficient computing package, `fastBLEU` (Alihosseini et al., 2019), to compute evaluation scores. Compared with other package, `fastBLEU` adopts the multi-threaded C++ implementation.

3 System Usage

In this section, we show a detailed guideline to use our system library. Users can run the existing models or add their own models as needed.

3.1 Running Existing Models

To run an existing model within TextBox, users only need to specify the dataset and model by setting hyper-parameters, *i.e.,* `dataset` and `model`. And then experiments can be run with a simple command-line interface:

```
python run_textbox.py \
  --model=GPT2 --dataset=COCO
```

The above case shows an example that runs GPT-2 (Radford et al., 2019) model on COCO dataset (Lin et al., 2015). In our system library, the generation task, such as `translation`, and `summarization`, is determined once users specify the dataset, thus the task is not necessary to be explicitly specified in hyper-parameters. To facilitate the modification of hyper-parameters, we provides two kinds of YAML configuration files, *i.e.,* dataset configuration and model configuration, which allow running many experiments without modifying source code. It also supports users to include hyper-parameters in the command line, which is useful for some specifically defined parameters. TextBox is designed to be run on different hardware devices. By default, CUDA devices will be used if users set the hyper-parameter `use_gpu` as `True`, or otherwise CPU will be used. Users can determine the ID of used CUDA devices by setting hyper-parameter `gpu_id`. We also support distributed model training in multiple GPUs by setting the hyper-parameter `DDP` as `True`.

Based on the configuration, we provide the auxiliary function to split the dataset into train, validation and test sets according to the provided hyper-parameter `split_ratio`, or load the pre-split dataset. Moreover, TextBox also allows users to load and re-train the saved model for speeding up reproduction, rather than training from scratch.

Figure 2 presents a general usage flow when running a model in our library. The running procedure relies on some experimental configuration, obtained from the files, command line or parameter dictionaries. The dataset and model are prepared and initialized according to the configured settings, and the execution module is responsible for training and evaluating models.

3.2 Implementing a New Model

With the unified `Data` and `Evaluation` modules, one needs to implement a specific `Model` class and three mandatory functions as follows:

- `__init__()` function. In this function, the user performs parameters initialization, global variable definition and so on. It is worth noting that, the imported new model should be a sub-class of the abstract model class defined in our library. One can

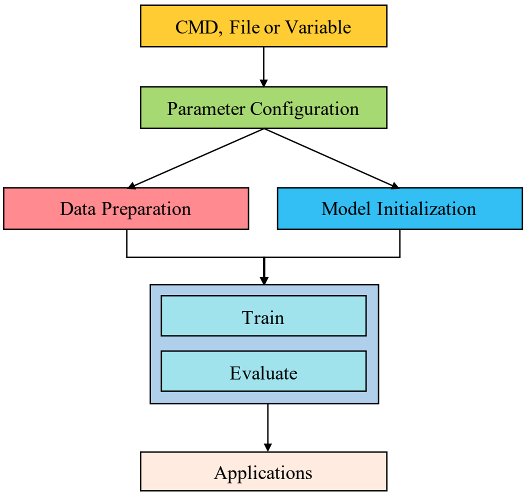

Figure 2: An illustractive usage flow of our library.

reuse the modules (*e.g.,* Transformer) and layers (*e.g.,* Highway net) already existing in our library for convenience. A configuration file is preferable to conduct further flexible adjustment.

• `forward()` function. This function calculates the training loss to be optimized and validation loss to avoid overfitting. Based on the returned training loss, our library will automatically invoke different optimization methods to learn the parameters according to pre-defined configuration.

• `generate()` function. This function is employed to generate output text based on input text or free text. Our library also provides several generation strategies, such as beam search and top-k search, for users to improve generation results.

In order to implement user-customized modules, one can reuse functions and classes inherited from our basic modules, or override original functions and add new functions.

4 Performance Evaluation

To evaluate the models in TextBox, we conduct extensive experiments to compare their performance on unconditional and conditional generation tasks.

4.1 Unconditional Text Generation

Following previous work, we adopt COCO (Lin et al., 2015), EMNLP2017 WMT News (Chatterjee et al., 2017) and IMDB Movie Reviews (Maas et al., 2011) datasets for comparing the performance of five traditional and state-of-the-art models, *i.e.,* LSTM-VAE, SeqGAN, RankGAN, MaliGAN, and GPT-2, in the unconditional text generation task.

In our experiments, we run models with the parameter configurations described in their original papers. Note that the BLEU-n metric employs the one-hot weights (*e.g.,* $(0, 0, 0, 1)$ for BLEU-4) instead of average weights, since we consider that one-hot weights can reflect the overlapping n-grams more realistically.

These results on COCO datasets are shown in Table 2, and other results on EMNLP2017 and IMDB datasets can be found in our GitHub page. We can see from Table 2, these models implemented in our library have the comparable performance compared with the results reported in the original papers. Moreover, the pretrained language model, *i.e.,* GPT-2, achieves consistent and remarkable performance, which is in line with our expectations.

4.2 Conditional Text Generation

In this section, we apply various models on four conditional text generation tasks, *i.e.,* attribute-to-text generation, dialogue systems, machine translation, and text summarization. The task of attribute-to-text generation is to generate text given several discrete attributes, such as user, item, and rating. We use the popular context-to-sequence (Context2Seq) and attribute-to-sequence (Attr2Seq) as base models, which utilize the multi-layer perceptron (MLP) and RNN as the encoder and decoder, respectively. Besides, dialogue systems aim to generate response given a conversation history. We consider two typical models, *i.e.,* attention-based RNN and Transformer, and one popular hierarchical recurrent encoder-decoder model (HRED) as base models. In RNN and Transformer, the multi-sequence conversation history is concatenated as one sequence feeding into the encoder, while in HERD the hierarchical structure of the conversation history is kept and modeled with a hierarchical encoder. Their results are shown in Table 2.

To showcase how our TextBox can support diverse techniques on several tasks with different decoding strategies, we compare the attention-based RNN model, Transformer, and four state-of-the-art pretrained language models, *i.e.,* BART, BERT2BERT, ProphetNet, and T5, for both machine translation and text summarization tasks. In Table 3, we adopt the IWSLT2014 German-to-English (Cettolo et al., 2014) translation dataset and utilize three generation strategies, *i.e.,* top-k, greedy, and beam search. The greedy strategy considers the most probable token at each generation step, the top-k search strategy means sorting by probability and zero-ing out the probabili-

Tasks	Datasets	Models	Distinct-1	Distinct-2	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Unconditional Generation	COCO	LSTM-VAE	-	-	63.97	46.56	18.53	5.97
		SeqGAN	-	-	99.76	82.32	51.26	25.18
		RankGAN	-	-	99.76	82.92	52.46	26.40
		MailGAN	-	-	99.71	81.95	50.86	24.87
		GPT-2	-	-	88.15	78.13	55.81	31.88
Attribute-to-Text Generation	AMAZON	Context2Seq	0.07	0.39	17.21	2.80	0.83	0.43
		Attr2Seq	0.14	2.81	17.14	2.81	0.87	0.48
Dialogue Systems	Personal Chat	RNN+Attn	0.24	0.72	17.51	4.65	2.11	1.47
		Transformer	0.38	2.28	17.29	4.85	2.32	1.65
		HRED	0.22	0.63	17.29	4.72	2.20	1.60

Table 2: Performance comparisons of different methods for three tasks, *i.e.*, unconditional generation, attribute-to-text generation, and dialogue systems. Distinct-n is not applicable to the unconditional generation task. "-" denotes the metric Distinct-n is generally not applicable to unconditional text generation.

Model	Strategy	BLEU2	BLEU3	BLEU4
RNN+Attn	**Top-k**	26.68	16.95	10.85
	Greedy	33.74	23.03	15.79
	Beam	35.68	24.94	17.42
Transformer	**Top-k**	30.96	20.83	14.16
	Greedy	35.48	24.76	17.41
	Beam	36.88	26.10	18.54

Table 3: Performance comparison of different generation models with three strategies for machine translation from German to English.

Model	ROUGE-1	ROUGE-2	ROUGE-L
RNN+Attn	36.32	17.63	38.36
Transformer	36.21	17.64	38.10
BART	39.34	20.07	41.25
BERT2BERT	38.16	18.89	40.06
ProphetNet	38.49	18.41	39.84
T5	38.83	19.68	40.76

Table 4: Performance comparison of different generation models for text summarization. Specifically, we adopt the base version of BART, BERT2BERT, T5 and the large version of ProphetNet.

ties for anything below the k-th token, and beam search (Vijayakumar et al., 2018) strategy selects the top scoring B candidates from the set of all possible one token extensions of its beams, where B is the beam size ($B = 5$ in our experiments). From Table 3 we observe that the beam search strategy brings more improvement than the others. For text summarization, we compare RNN and Transformer with four pretrained models as shown in Table 4. These models are trained or fine-tuned in GigaWord (Graff et al., 2003) dataset. As observed in Table 4, pretrained models outperform the RNN model and Transformer by a clear margin.

The results of all implemented models in other tasks can be acquired from our GitHub page.

5 Related Work

Several toolkits have been released focusing on one or a few specific text generation tasks or techniques. For example, Tensor2Tensor (Vaswani et al., 2018), MarianNMT (Junczys-Dowmunt et al., 2018) and OpenNMT (Klein et al., 2017a) are designed for machine translation task, while ParlAI (Miller et al., 2017a) and Plato (Papangelis et al., 2020) specialized for dialog research in this field. There are two text generation libraries closely related to our library, including Texygen (Zhu et al., 2018) and Texar (Hu et al., 2019) focusing on GAN technique and high modularization, respectively. TextBox has drawn inspirations from these toolkits when designing relevant functions.

Compared with them, TextBox covers more text generation tasks and models, which is useful for reproducibility. Besides, we implement standardized evaluation to compare different models. Also, our library provides various common modules for convenience. It has a proper focus on text generation field, and provide a comprehensive set of modules and functionalities.

6 Conclusion

This paper presented a unified, modularized, and extensible text generation library, called `TextBox`. So far, we have implemented 21 text generation models, including VAE-based, GAN-based, pretrained language models, sequence-to-sequence and 9 benchmark datasets for unconditional and conditional text generation tasks. Moreover, Our library is modularized to easily plug in or swap out components, and extensible to support seamless incorporation of other external modules. In the future, features and functionalities will continue be added to our library, including more models and datasets, diverse inputs such as graph and table, and distributed training in multiple machines. We invite researchers and practitioners to join and enrich TextBox, and help push forward the research on text generation.

7 Broader Impacts

Text generation has a wide range of beneficial applications for society, including code auto-completion, game narrative generation, and answering questions. But it also has potentially harmful applications. For example, GPT-3 improves the quality of generated text over smaller models and increases the difficulty of distinguishing synthetic text from human-written text, such as fake news and reviews.

Here we focus on two potential issues: the potential for deliberate misuse of generation models and the issue of bias. Malicious uses of generation models can be somewhat difficult to anticipate because they often involve repurposing models in a very different environment or for a different purpose than researchers intended. To mitigate this, we can think in terms of traditional security risk assessment frameworks such as identifying threats. Biases present in training text may lead models to generate stereotyped or prejudiced content. This is concerning, since model bias could harm people in the relevant groups in different ways. In order to prevent bias, there is a need for building a common vocabulary tying together the normative, technical and empirical challenges of bias mitigation for generation models. We expect this to be an area of continuous research for us.

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China under Grant No. 61872369 and 61832017, Beijing Academy of Artificial Intelligence (BAAI), Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098, the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China under Grant No. 18XNLG22 and 19XNQ047. Xin Zhao is the corresponding author.

References

Danial Alihosseini, Ehsan Montahaei, and Mahdieh Soleymani Baghshah. 2019. Jointly measuring diversity and quality in text generation models. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 90–98, Minneapolis, Minnesota. Association for Computational Linguistics.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *IEEvaluation@ACL*, pages 65–72. Association for Computational Linguistics.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 10–21.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1442–1451, Copenhagen, Denmark. Association for Computational Linguistics.

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, Jennifer C. Lai, and Robert L. Mercer. 1992. An estimate of an upper bound for the entropy of english. *Comput. Linguistics*, 18(1):31–40.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, volume 57.

Rajen Chatterjee, Matteo Negri, Marco Turchi, Marcello Federico, Lucia Specia, and Frédéric Blain. 2017. Guiding neural machine translation decoding with external knowledge. In *Proceedings of the Second Conference on Machine Translation, Volume 1: Research Papers*, pages 157–168, Copenhagen, Denmark. Association for Computational Linguistics.

Tong Che, Yanran Li, Ruixiang Zhang, R. Devon Hjelm, Wenjie Li, Yangqiu Song, and Yoshua Bengio. 2017. Maximum-likelihood augmented discrete generative adversarial networks. *arXiv preprint arXiv:1702.07983*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, pages 4171–4186. Association for Computational Linguistics.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 623–632. Association for Computational Linguistics.

William Fedus, Ian J. Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the _______. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 5141–5148.

Zhiting Hu, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He, Lianhui Qin, Di Wang, Xuezhe Ma, Zhengzhong Liu, Xiaodan Liang, Wanrong Zhu, Devendra Singh Sachan, and Eric P. Xing. 2019. Texar: A modularized, versatile, and extensible toolkit for text generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28 - August 2, 2019, Volume 3: System Demonstrations*, pages 159–164. Association for Computational Linguistics.

Ferenc Huszar. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017a. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017b. Opennmt: Open-source toolkit for neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, System Demonstrations*, pages 67–72. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 110–119. The Association for Computational Linguistics.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1192–1202. The Association for Computational Linguistics.

Juntao Li, Yan Song, Haisong Zhang, Dongmin Chen, Shuming Shi, Dongyan Zhao, and Rui Yan. 2018. Generating classical chinese poems via conditional variational autoencoder and adversarial training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3890–3900. Association for Computational Linguistics.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2021a. Few-shot knowledge graph-to-text generation with pre-trained language models. In *Findings of ACL*.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021b. Pretrained language models for text generation: A survey. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI 2021*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Kevin Lin, Dianqi Li, Xiaodong He, Ming-Ting Sun, and Zhengyou Zhang. 2017. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*

2017, December 4-9, 2017, Long Beach, CA, USA, pages 3155–3165.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. 2015. Microsoft coco: Common objects in context.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Alexander Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017a. ParlAI: A dialog research software platform. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 79–84, Copenhagen, Denmark. Association for Computational Linguistics.

Alexander H. Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017b. Parlai: A dialog research software platform. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017 - System Demonstrations*, pages 79–84. Association for Computational Linguistics.

Alexandros Papangelis, Mahdi Namazifar, Chandra Khatri, Yi-Chia Wang, Piero Molino, and Gokhan Tur. 2020. Plato dialogue system: A flexible conversational ai research platform.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Maja Popovic. 2015. chrf: character n-gram f-score for automatic MT evaluation. In *WMT@EMNLP*, pages 392–395. The Association for Computer Linguistics.

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, EMNLP 2020, Online Event, 16-20 November 2020*, pages 2401–2410. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Trans. Assoc. Comput. Linguistics*, 8:264–280.

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, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1073–1083. Association for Computational Linguistics.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 627–637.

Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 3776–3784. AAAI Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Jian Tang, Yifan Yang, Samuel Carton, Ming Zhang, and Qiaozhu Mei. 2016. Context-aware natural language generation with recurrent neural networks. *arXiv preprint arXiv:1611.09900*.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2Tensor for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 193–199, Boston, MA. Association for Machine Translation in the Americas.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575. IEEE Computer Society.

Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 7371–7379. AAAI Press.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 3881–3890.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 2852–2858.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *ICLR*. OpenReview.net.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. 2017. Adversarial feature matching for text generation. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 4006–4015.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1097–1100. ACM.