TextBox 2.0: A Text Generation Library with Pre-trained Language Models

Tianyi Tang\textsuperscript{1,5†}, Junyi Li\textsuperscript{1,3†}, Zhipeng Chen\textsuperscript{2†}, Yiwen Hu\textsuperscript{2}, Zhuohao Yu\textsuperscript{2}, Wenzxun Dai\textsuperscript{4}, Wayne Xin Zhao\textsuperscript{1,5†}, Jian-Yun Nie\textsuperscript{3}, and Ji-Rong Wen\textsuperscript{1,2,5}

\textsuperscript{1}Gaoling School of Artificial Intelligence, Renmin University of China \textsuperscript{2}School of Information, Renmin University of China \textsuperscript{3}DIRO, Université de Montréal \textsuperscript{4}Xidian University
\textsuperscript{5}Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE

\texttt{steventianyitang@outlook.com lijunyi@ruc.edu.cn batmanfly@gmail.com}

Abstract

To facilitate research on text generation, this paper presents a comprehensive and unified library, TextBox 2.0, focusing on the use of pre-trained language models (PLMs). To be comprehensive, our library covers 13 common text generation tasks and their corresponding 83 datasets and further incorporates 45 PLMs covering general, translation, Chinese, dialogue, controllable, distilled, prompting, and lightweight PLMs. We also implement 4 efficient training strategies and provide 4 generation objectives for pre-training new PLMs from scratch. To be unified, we design the interfaces to support the entire research pipeline (from data loading to training and evaluation), ensuring that each step can be fulfilled in a unified way. Despite the rich functionality, it is easy to use our library, either through the friendly Python API or command line. To validate the effectiveness of our library, we conduct extensive experiments and exemplify four types of research scenarios. The project is released at the link: https://github.com/RUCAIBox/TextBox\textsuperscript{2.0}.

1 Introduction

Text generation, aiming to generate human-like texts on demand, has been a fundamental technique in many text applications, such as machine translation (Dabre et al., 2020), text summarization (El-Kassas et al., 2021), and dialogue system (Chen et al., 2017). Recently, pre-trained language models (PLMs) such as BART (Lewis et al., 2020) have been the mainstream approach to developing effective text generation models. With the great advances in text generation, it has become increasingly important to reproduce, develop, and compare various text generation models in a reliable, flexible, and unified way.

Considering the rapid progress of PLMs on text generation, in this paper, we present a significant extension of a previously released text generation library, TextBox 1.0 (Li et al., 2021), called TextBox 2.0. Different from TextBox 1.0 and other text generation libraries (Miller et al., 2017; Klein et al., 2018; Zhu et al., 2018) (mostly including classical models based on recurrent neural networks or generative adversarial networks), this extension mainly focuses on building a comprehensive and unified framework for better supporting PLM-based text generation models. Although some libraries (e.g., Fairseq (Ott et al., 2019) and Hugging Face (Wolf et al., 2020)) also include PLMs, they are designed for performing myriad NLP tasks (only considering a few text generation tasks). Moreover, they don’t maintain a complete evaluation pipeline (e.g., data loading, training, inference, and evaluation) specially designed for text generation. Thus, it is not fully suited for developing and evaluating text generation models in a unified way.

In order to better facilitate research on text generation, TextBox 2.0 introduces a series of new features for supporting the use of PLMs, which can be summarized into three major aspects:

- **Generation Tasks**: Our library supports 13 commonly studied text generation tasks (e.g., translation and story generation) and their corresponding 83 datasets, including most of the existing mainstream tasks and datasets for research. We reorganize these datasets so that they are framed in a unified text-to-text format. Users can simply set the dataset via the command line or configuration file without additional preprocessing efforts.

- **Generation Models**: As a key contribution, our library incorporates 45 PLMs, covering the categories of general, translation, Chinese, dialogue, controllable, distilled, prompting, and lightweight PLMs. We unify the interface to use existing PLMs and incorporate new PLMs, and it is convenient to run different PLMs for a specified task in our
library. We also provide a standard way to compare these models and analyze the generated results.

- **Training Strategies:** To support the optimization of PLMs, we provide four efficient and robust training strategies (e.g., efficient decoding) and four pre-training objectives (e.g., denoising auto-encoding) for text generation. These strategies make optimizing text generation models more efficient and reliable. Users can either pre-train a new model from scratch or fine-tune a pre-trained model for research purposes.

As another merit, TextBox 2.0 has been largely aligned with our previous survey on PLM-based text generation (Li et al., 2022b) in terms of task, model, and training. It will be meaningful for beginners to explore and learn text generation models with the survey and supporting libraries.

To summarize, TextBox 2.0 has contributed a significant addition to the previous version (see Table 1 for a detailed comparison) to better support the use of PLMs for text generation. It implements and maintains a unified way to conduct research on text generation with 45 included models, covering 13 tasks, and 83 datasets. We also perform extensive test experiments, and these results show that TextBox 2.0 can produce very competitive performance compared to the original implementations.

<table>
<thead>
<tr>
<th>Aspects</th>
<th>TextBox 1.0</th>
<th>TextBox 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
<td>6 v.s. 13</td>
<td>Summarization, translation, dialogue, unconditional generation, attribute-to-text generation, poem generation</td>
</tr>
<tr>
<td>Models</td>
<td>6 v.s. 45</td>
<td>General: GPT-2, BERT2BERT, BART, T5, ProphetNet, GPT, GPT-Neo, OPT, UniLM, MASS, PEGASUS, MVP, Bigbird, LED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Translation: mBART, mT5, Marian, M2M 100, NLLB, XLM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chinese: CPM, CPT, Chinese-BART, Chinese-T5, Chinese-GPT2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dialogue: Blenderbot and DiaLoGPT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Controllable: CTRL and PPLM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Distilled: DistilGPT2 and DistilBART</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Prompting: PTG and Context-Tuning</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lightweight: Adapter, Prefix-tuning, Prompt tuning, LoRA, BitFit, P-Tuning v2</td>
</tr>
<tr>
<td>Training Strategies</td>
<td>Distributed data parallel</td>
<td>Distributed data parallel, efficient decoding, hyper-parameter optimization, repeated experiments, pre-training objectives</td>
</tr>
</tbody>
</table>

Table 1: Comparison of TextBox 1.0 and TextBox 2.0. We also present a comparison of the numbers of tasks and pre-trained models between the two versions.

2.1 Generation Tasks

Since there are various text generation applications, we include 13 widely studied tasks and collect the corresponding 83 datasets.

- **Tasks.** These 13 tasks in TextBox 2.0 include text summarization, machine translation, open-ended dialogue system, data-to-text generation, question generation, question answering, story generation, task-oriented dialogue system, commonsense generation, paraphrase generation, text style transfer, and text simplification. Besides these English-centric tasks, we also include Chinese generation tasks. Existing PLM-based libraries such as Hugging Face (Wolf et al., 2020) are focused on performing extensive NLP tasks and only consider a few text generation tasks (mainly text summarization and machine translation), which are not comprehensive for text generation research.

- **Datasets.** For each task, we collect widely-used datasets and reorganize them in a unified text-to-text format. In total, we include 83 datasets, and report their details on the page\(^1\), including the dataset description, basic statistics, and training/validation/testing samples. In addition, we build a leaderboard for each dataset by collecting the automatic results and generated texts of the latest research. It is convenient for users to quickly learn about the baselines and their results. We also encourage community users to collaboratively maintain the leaderboard and submit their model results.

\(^1\)https://github.com/RUCAIBox/TextBox#dataset
Metrics. To conduct evaluations with these tasks and datasets, TextBox 2.0 supports four categories of automatic metrics: (1) lexical metrics, such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004), to measure the n-gram overlap between generated texts and golden texts; (2) semantic metrics, such as BERTScore (Zhang et al., 2020b) and style strength (Lai et al., 2021), to compare the texts at sentence level; (3) diversity metrics, such as Distinct (Li et al., 2016) and Self-BLEU (Zhu et al., 2018), to evaluate the lexical diversity of generated texts; (4) accuracy metrics, such as exact match (Rajpurkar et al., 2016) and inform (Budzianowski et al., 2018a), to calculate the precision of important phrases. In total, we include 12 general metrics and 5 task-specific metrics.

Besides the analysis using automatic metrics, TextBox 2.0 provides several visualization tools to explore and analyze the generated texts in various dimensions (Liu et al., 2021b; Tuckute et al., 2022). For instance, Figure 2 shows how it offers new insights to improve summarization tasks (details can be found in Section 4.3).

2.2 Generation Models
To support the rapid progress of PLMs on text generation, TextBox 2.0 incorporates 45 PLMs and aims to build a unified and standardized framework based on PLMs. We list some included models as follows:

- **General PLMs**: GPT-2 (Radford et al., 2019) and BART (Lewis et al., 2020);
- **Translation PLMs**: mBART (Liu et al., 2020) and XLM (CONNEAU and Lample, 2019);
- **Chinese PLMs**: CPM (Zhang et al., 2021) and CPT (Shao et al., 2021);
- **Dialogue PLMs**: DialoGPT (Zhang et al., 2020c) and Blenderbot (Roller et al., 2021);
- **Controllable PLMs**: CTRL (Keskar et al., 2019) and PPLM (Dathathri et al., 2020);
- **Distilled PLMs**: DistilGPT2 (Sanh et al., 2019) and DistilBART (Shleifer and Rush, 2020);
- **Prompting PLMs**: PTG (Li et al., 2022a) and Context-Tuning (Tang et al., 2022);
- **Lightweight modules**: Adapter (Houlsby et al., 2019), Prefix-tuning (Li and Liang, 2021).

The wide coverage of PLMs makes it possible to deal with different text generation tasks using TextBox 2.0. For example, to perform specific tasks such as dialogue system, users can adopt task-specific PLMs such as DialoGPT; to deal with Chinese generation tasks, users can adopt CPT. In resource-constrained situations, lightweight PLMs such as prefix-tuning can be a good choice.

2.3 Training Strategies
TextBox 2.0 provides four pre-training objectives to help users pre-train a model from scratch, including language modeling (Radford et al., 2019), masked sequence-to-sequence modeling (Song et al., 2019), denoising auto-encoding (Lewis et al., 2020), and masked span prediction (Raffel et al., 2020). These pre-training tasks can also be utilized for domain-adaptive pre-training and task-adaptive pre-training (Gururangan et al., 2020) to tailor existing PLM to the domain of a target task.

Also, TextBox 2.0 provides four useful training methods for improving the optimization of PLMs. It supports distributed data parallel to implement models on multiple GPUs and machines to improve the efficiency of pre-training and fine-tuning. We incorporate Accelerate to support distributed training with a simple API. To further accelerate the decoding efficiency, we integrate FastSeq (Yan et al., 2021) to optimize the decoding process by attention cache optimization, repeated n-gram detection, and asynchronous parallel I/O.

Moreover, TextBox 2.0 enables users to adjust and select hyper-parameters automatically. Based on the library Hyperopt (Bergstra et al., 2013), users just need to set the parameter range and search methods, and then the optimal hyper-parameters and corresponding results will return. It is useful for PLMs to search for hyper-parameters such as batch size and learning rate. Our library also supports performing repeat experiments using different random seeds in one command line, which is especially useful to alleviate randomness especially under few-shot settings.

3 Library Usage
In this section, we introduce how to use our library in four different kinds of research scenarios by showing the example codes.

Reproducing existing models. TextBox 2.0 includes various PLMs and supports many text generation tasks and datasets. It is convenient for users to reproduce existing models. For example, to run the GPT-2 model, users can simply run the following command:

```bash
python run_gpt2.py --model_name_or_path gpt2
```

This command will download the GPT-2 model and run it on a text generation task.

4https://github.com/huggingface/accelerate
to quickly run existing PLMs and reproduce results for each dataset. In particular, users only need to specify the dataset and model by setting the configurations dataset, model, and model_path, within a simple command line.

Figure 1(a) presents an example to fine-tune PEGASUS (Zhang et al., 2020a) on XSum (Narayan et al., 2018) dataset. Moreover, TextBox 2.0 enables users to conduct hyper-parameter optimization by only providing a list of possible values. Figure 1(b) shows an example that automatically adjusts the hyper-parameters learning_rate and batch_size from the ranges [1 × 10^{-5}, 3 × 10^{-5}] and [64, 256], respectively.

**Implementing a new model.** Since TextBox 2.0 builds a unified pipeline for text generation research, users only need to define a new model class without considering other procedures to implement a new model. Specially, users should first inherit from our base model class AbstractModel before specifying three specific model functions: (1) __init__(self): this function initializes the architectures and parameters of the model; (2) forward(self, input, label): this function is used to calculate the loss for optimization during training; (3) generate(self, input): this function generates texts based on input during inference.

Figure 1(c) presents an example of implementing a new model for the KG-to-text generation task. In this example, the model adopts a graph neural network (GNN) to encode KG and then uses T5 (Raffel et al., 2020) to generate texts. We first define the GNN and T5 models in the __init__ function. Then, we use GNN to encode KG to embeddings as the input of T5 and compute the loss according to target labels in the forward function. Finally, we use a similar process to generate text in the generate function.

**Pre-training a new model.** In TextBox 2.0, we provide several pre-training objectives for users to pre-train new models from scratch. Specifically, users just need to specify the pre-training task, pre-training corpus, and architecture by setting pretrain_task, dataset, and model. Figure 1(d) shows an example that pre-trains a Chinese BART on the WuDaoCorpora (Yuan et al., 2021) using the denoising pre-training objective.

To improve the pre-training efficiency, TextBox 2.0 supports distributed data parallel and efficient decoding (Section 2.3). Figure 1(e) shows an illustrative example of how users can use the accelerate command to set configurations of multiple devices and launch the training code.

**Analyzing generated results.** Besides simply obtaining the evaluation results, our library provides several visualization analysis mechanisms to perform deep analysis on the generated results of models. For example, we support the use of the statistical chart to analyze the mean and standard deviation scores for different sentence lengths. These methods can help users learn about the advantages and disadvantages of different models in a detailed comparison. Figure 1(f) shows an example of how to run the analysis using a simple command line and the results can be found in Figure 2. This example compares the generated texts of BART and T5 on the CNN/DailyMail dataset.

4 Experiments

In this section, we conduct extensive experiments to verify the generation abilities of TextBox 2.0.

4.1 Result Reproduction

As an open-source library, TextBox 2.0 should be able to reproduce the results of existing work effectively. To verify this, we select a number of
R-1 R-2 R-L B-4 ME R-2 LCSTS CSL. ADGEN En→Ro Ro→En
BART 44.16 44.47 40.90 83.30 55.60 86.10 40.60 64.20 10.00 37.70 37.80
BART (ours) 4 4 4 4 4 4 4 4 4 4 4 4

Data-to-text Generation Commonsense Generation Question Generation QA
B-4 ME R-L B-4 CIDEr SPICE B-4 ME R-L B-4 F1 EM
BART 64.55 46.51 75.13 27.50 14.12 30.00 22.00 26.40 50.30 1.56 84.23
BART (ours) 67.33 47.78 76.83 28.18 12.98 33.00 25.08 26.73 52.55 1.56 86.44

Open-ended Dialogue System Task-oriented Dialogue System Story Generation
B-1 B-2 D-1 D-2 B-4 Success Inform Comb. B-1 B-2 D-4
BART 49.90 40.00 1.30 8.00 17.89 74.91 84.88 97.80 30.70 13.30 69.90
BART (ours) 49.58 39.24 1.44 8.89 20.17 75.40 84.40 100.07 33.79 15.78 78.76

Table 2: The results of BART on thirteen tasks from the original papers and our TextBox 2.0. QA is short for question answering. B, R, D, ME, EM, HM, Acc., and Comb. denote BLEU, ROUGE, Diverse, METEOR, exact match, harmonic mean, mean, and accuracy, and combined score, respectively. LCSTS, CSL, ADGEN, and En++Ro are evaluated using the R-L, R-L, B-4, and B-4 metrics, respectively. ①(Lewis et al., 2020) ②(Kehrmann et al., 2021) ③(Shao et al., 2021) ④(Liu et al., 2020) ⑤(Ke et al., 2021) ⑥(Lin et al., 2020a) ⑦(Liu et al., 2021a) ⑧(Xu et al., 2021) ⑨(Lin et al., 2020b) ⑩(Guan et al., 2021) ⑪(Sun et al., 2021) ⑫(Lai et al., 2021)

Library Preparation Training Generation
Preparation (minutes) Training (minutes) Generation (minutes)
Fairseq 2.93 410.05 79.24 95.3
Hugging Face 4.02 416.25 75.69 55.5
TextBox 2.0 3.81 393.99 27.05 65.1

Table 3: Efficiency comparison of three libraries for BART\textsubscript{LARGE} fine-tuned on CNN/DailyMail. The preparation stage consists of configuration loading, text tokenization, and necessary initialization options. The training stage takes time for fine-tuning on the training set in one epoch. The generation stage takes time to generate on the test set with a beam size of 5.

widely-used datasets for each task (introduced in Section 2.1) and compare the results conducted by TextBox 2.0 with those in the original papers. We totally evaluate 13 tasks using 14 datasets, including CNN/DailyMail (See et al., 2017), WikiAuto + Turk (Liu et al., 2021a), LCSTS (Hu et al., 2015), CSL\textsuperscript{5}, ADGEN (Shao et al., 2019), WMT 16 English-Romanian (En++Ro) (Bojar et al., 2016), WebNLG 2.1 (Gardent et al., 2017), CommonGen (Lin et al., 2020a), SQuAD (Rajpurkar et al., 2016), PersonaChat (Zhang et al., 2018), MultiWOZ 2.0 (Budzianowski et al., 2018b), ROCStories (Mostafazadeh et al., 2016), GYAFC (E&M and F&R) (Rao and Tetreault, 2018), and Quora (Kumar et al., 2020).

Since BART is the prevalent PLM for text generation, we endeavor to reproduce existing works with BART\textsubscript{LARGE}.\textsuperscript{6} For all experiments, we employ the sequence-to-sequence cross-entropy loss with a label smoothing factor of 0.1 as the objective function. We optimize the model using AdamW (Loshchilov and Hutter, 2019) with a constant learning rate of 3 × 10\textsuperscript{−5}. The accumulated batch size is set to 192. During inference, we apply beam search with a beam size of 5 and no-repeat

\textsuperscript{5}https://github.com/CLUEbenchmark/CLGE
\textsuperscript{6}For translation tasks, we utilize mBART-CC25 (Liu et al., 2020). For Chinese generation tasks, we utilize Chinese BART\textsubscript{LARGE} (Shao et al., 2021).
4.2 Efficiency Comparison

In addition to accurately reproducing results, we have optimized TextBox 2.0 for computational efficiency. We streamline the training process and support efficient decoding strategies. To compare the efficiency, we choose the well-known PLM libraries Fairseq\(^7\) and Hugging Face\(^8\), and then test the time consumption under identical settings described in Section 4.1.

From the results in Table 3, we can see that our TextBox 2.0 is more efficient than Fairseq and Hugging Face. During training, TextBox 2.0 simplifies the training process and reduces the time spent on non-essential functions such as trainer management and loss tracking. In the generation process, our library is significantly faster than the other two libraries due to the incorporation of efficient decoding strategies introduced in Section 2.3.

4.3 Visualization Analysis

Besides reproducing a model, it is also important to compare existing methods, analyze the generated texts, and explore directions for improvement. Our library sets a specific leaderboard for each dataset, including basic metric results, author repositories, and generated texts. Figure 2 (a) showcases the leaderboard for the CNN/DailyMail dataset.

Users can also utilize TextBox 2.0 to conduct visualization analysis for specified models. For example, our library can automatically plot the boxplot of the ROUGE-L score for different input lengths and the \(n\)-gram overlap of target and generated texts with the input document. From the results in Figure 2 (b), we can find that T5 excels at short document summarization while BART excels at long document summarization. It is useful to analyze and improve the deficiencies of text generation models or obtain better performance by combining their results. As another example, Figure 2 (c) illustrates that BART and T5 have a significantly higher \(n\)-gram overlap ratio than golden sentences, indicating that they tend to “copy” the input document rather than “summarize” it. From such analysis results, users can apply the methods proposed by Goyal et al. (2022) to alleviate it.

5 Conclusion

This paper presented TextBox 2.0, a comprehensive and unified library for conducting research on PLM-based text generation. Our library makes significant extensions in three major aspects, namely generation tasks (13 tasks and 83 datasets), generation models (45 PLMs), and training strategies (e.g., distributed data parallel and efficient decoding). Results from extensive test experiments demonstrate that our library can accurately reproduce existing models. Besides, we also provide a series of utility tools to better analyze and explore the generated results. To summarize, our library can be very useful to facilitate text generation research, and our team will improve this library with regular updates.

Acknowledgement

This work was partially supported by Beijing Natural Science Foundation under Grant No. 4222027, and Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098. Xin Zhao is the corresponding author.
References


Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea


Dayiheng Liu, Yu Yan, Yeyun Gong, Weizhen Qi, Hang Zhang, Jian Jiao, Weizhu Chen, Jie Fu, Linjun Shou, Ming Gong, Pengcheng Wang, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, Ruofei Zhang, Winnie Wu, Ming Zhou, and Nan Duan. 2021a. GLGE: A


