

OpenPrompt: An Open-source Framework for Prompt-learning

Ning Ding^{*}, Shengding Hu^{*}, Weilin Zhao^{*}, Yulin Chen,
Zhiyuan Liu[†], Hai-Tao Zheng[†], Maosong Sun

Tsinghua University, Beijing, China

{dingn18, hsd20, zwl19, yl-chen21}@mails.tsinghua.edu.cn

Abstract

Prompt-learning has become a new paradigm in modern natural language processing, which directly adapts pre-trained language models (PLMs) to *cloze*-style prediction, autoregressive modeling, or sequence to sequence generation, resulting in promising performances on various tasks. However, no standard implementation framework of prompt-learning is proposed yet, and most existing prompt-learning codebases, often unregulated, only provide limited implementations for specific scenarios. Since there are many details such as templating strategy, initializing strategy, and verbalizing strategy, etc. need to be considered in prompt-learning, practitioners face impediments to quickly adapting the desired prompt learning methods to their applications. In this paper, we present OpenPrompt, a unified easy-to-use toolkit to conduct prompt-learning over PLMs. OpenPrompt is a research-friendly framework that is equipped with efficiency, modularity, and extensibility, and its combinability allows the freedom to combine different PLMs, task formats, and prompting modules in a unified paradigm. Users could expediently deploy prompt-learning frameworks and evaluate the generalization of them on different NLP tasks without constraints. OpenPrompt is publicly released at <https://github.com/thunlp/OpenPrompt>.

1 Introduction

Pre-trained language models (PLMs) (Han et al., 2021a; Qiu et al., 2020) have been widely proven to be effective in natural language understanding and generation, ushering in a new era of modern natural language processing (NLP). In the early stage of this revolution, a standard approach to adapt PLMs to various specific NLP tasks is the

pretraining-finetuning paradigm, where additional parameters and task-specific objectives are introduced in the tuning procedure. However recently, the paradigm of the adaptation of PLMs is shifting. Originated in T5 (Raffel et al., 2019) and GPT-3 (Brown et al., 2020), researchers find that PLMs can be effectively stimulated by textual prompts or demonstrations, especially in low-data scenarios.

Take a simple prompt-based sentiment classification for example, the pipeline consists of a template and a verbalizer, where a template is used to process the original text with some extra tokens, and a verbalizer projects original labels to words in the vocabulary for final prediction. Assume the template is “<text> It is <mask>”, where the token <text> stands for the original text, and the verbalizer is {“positive”:“great”, “negative”:“terrible”}. The sentence “Albert Einstein was one of the greatest intellects of his time.” will first be wrapped by the pre-defined template as “Albert Einstein was one of the greatest intellects of his time. It is <mask>”. The wrapped sentence is then tokenized and fed into a PLM to predict the distribution over vocabulary on the <mask> token position. It is expected that the word *great* should have a larger probability than *terrible*.

As illustrated above, prompt-learning projects the downstream tasks to pre-training objectives for PLMs with the help of textual or soft-encoding prompts. A series of studies of prompt-learning (Liu et al., 2021a) have been proposed to investigate the strategies of constructing templates (Schick and Schütze, 2021; Gao et al., 2021; Liu et al., 2021b), verbalizers (Hu et al., 2021), optimization (Lester et al., 2021), and application (Li and Liang, 2021; Han et al., 2021b; Ding et al., 2021a) for this paradigm.

A prompt-learning problem could be regarded as a synthesis of PLMs, human prior knowledge, and specific NLP tasks that need to be handled.

^{*} equal contribution

[†] corresponding authors

Example	PLM	Template	Verbalizer	Task	Reference
Naive TC	MLM & Seq2Seq	M. text	M. One-Many	Text Classification	-
Naive KP	LM & Seq2Seq	M. text	-	Knowledge Probing	-
Naive FET	MLM	M. text (meta info)	M. One-Many	Entity Typing	(Ding et al., 2021a)
PTR	MLM	M. text (complex)	M. One-One	Relation Extratcion	(Han et al., 2021b)
P-tuning	LM	Soft tokens	M. One-One	Text Classification	(Liu et al., 2021b)
Prefix-tuning	LM, Seq2Seq	Soft tokens	-	Text Generation	(Li and Liang, 2021)
LM-BFF	MLM	A. text	M. One-Many	Text Classification	(Gao et al., 2021)

Table 1: Some examples implemented by OpenPrompt, where M. is the abbreviation of manually defined and A. is the abbreviation of automatically generated. Note that different approaches focus on different parts in prompt-learning. Additional to the whole pipeline, our specific implementations of these methods are integrated into the specific classes of OpenPrompt. For example, the core implementation of KPT is in the KnowledgeableVerbalizer class.

Hence, it is hard to support the particular implementations of prompt-learning elegantly with the current deep learning or NLP libraries while there is also a lack of a standard paradigm. Previous works pursue the most efficient way to implement prompt-learning with the least modification to the existing framework for traditional fine-tuning, resulting in poor readability and even unstable reproducibility. Moreover, the performance of a prompt-learning pipeline varies greatly with the choice of templates and verbalizers (Zhao et al., 2021), creating more barriers for implementations. Lastly, there is no comprehensive open-source framework particularly designed for prompt-learning at present, which makes it difficult to try out new methods and make rigorous comparisons for previous approaches.

To this end, we present OpenPrompt, an open-source, easy-to-use, and extensible toolkit for prompt-learning. OpenPrompt modularizes the whole framework of prompt-learning and considers the interactions between each module. We highlight the feature of combinability of OpenPrompt, which supports flexible combinations of diverse task formats, PLMs, and prompting modules. For example, we can easily adapt prefix-tuning (Li and Liang, 2021) to a text classification task in OpenPrompt. This feature enables users to assess the generalization of their prompt-learning models on various tasks, but not only the performance on specific tasks.

Specifically, in OpenPrompt, a Template class is used to define or generate textual or soft-encoding templates to wrap the original input. To flexibly support various templates under a unified paradigm, we design a new template language that could easily conduct token-level customization for the corresponding attributes. For example, users can specify which tokens are shared embedding,

trainable, or in what way these tokens are to be post-processed, without having to perform complex implementations for specific templates. A Verbalizer projects the classification labels to words in the vocabulary, and a PromptModel is responsible for the training and inference process. Each module in OpenPrompt is clearly defined while retaining its independence and coupling so that researchers can easily deploy a model and make targeted improvements. We also implement baselines with OpenPrompt and evaluate them on a broad scope of NLP tasks, demonstrating the effectiveness of OpenPrompt.

The area of prompt-learning is in the exploratory stage with rapid development. Hopefully, OpenPrompt could help beginners quickly understand prompt-learning, enable researchers to efficiently deploy prompt-learning research pipeline, and empower engineers to readily apply prompt-learning to practical NLP systems to solve real-world problems. OpenPrompt will not only open source all the code, but will also continue to update the documentation to provide detailed tutorials.

2 Background

Prompt-learning reveals what the next generation of NLP may look like.

Although PLMs have achieved tremendous success on almost all the subtasks in NLP, one problem still hangs in the air, *have we really fully exploited the potential of PLMs, especially the big ones?* Conventional fine-tuning uses extra task-specific heads and objectives for adaptation, but this strategy may face two issues. On the one hand, such an approach creates a natural gap between model tuning and pre-training. On the other hand, as the number of model parameters increases, this fine-tuning approach becomes increasingly difficult to

operate due to the massive computational volume (e.g., GPT-3 (Brown et al., 2020)).

By mimicking the process of pre-training, prompt-learning intuitively bridges the gap between pre-training and model tuning. Practically, this paradigm is surprisingly effective in low-data regime (Le Scao and Rush, 2021; Gao et al., 2021). For example, with appropriate template, zero-shot prompt-learning could even outperform 32-shot fine-tuning (Ding et al., 2021a). Another promising empirical attribute of prompt-learning is the potential to stimulate large-scale PLMs. When it comes to a 10B model, solely optimizing prompts (the parameters of the model are fixed) could achieve comparable performance to full parameter fine-tuning (Lester et al., 2021). These practical studies imply that we may use prompts to more effectively and efficiently dig the knowledge kept in PLMs, leading to a deeper understanding of the underlying principles of their mechanisms (Wei et al., 2021; Qin et al., 2021; Vu et al., 2021).

From a practical implementation point of view, prompt-learning is actually complex and requires a lot of detailed consideration. With general-purpose NLP under the prompt-learning paradigm as our target, we present OpenPrompt, a unified toolkit to effectively and efficiently implement prompt-learning approaches. OpenPrompt demonstrates a comprehensive view of the programming details of prompt-learning, and enables practitioners to quickly understand the mechanisms and practical attributes of this technique. And one can quickly deploy existing representative prompt-learning algorithms that are already implemented in the package under a unified programming framework. Moreover, OpenPrompt allows researchers or developers to quickly try out new ideas of prompt-learning, which not only includes newly designed templates or verbalizers, but also the exploration of the attributes of prompt-learning, e.g., prompt-based adversarial attacking.

3 Design and Implementation

As stated in § 1, prompt-learning is a comprehensive process that combines PLMs, human knowledge, and specific NLP tasks. Keeping that in mind, the design philosophy is to simultaneously consider the independence and mutual coupling of each module. As illustrated in Figure 1, OpenPrompt provides the full life-cycle of prompt-learning based on PyTorch (Paszke et al., 2019). In this section, we

first introduce the combinability of OpenPrompt, and then the detailed design and implementation of each component in OpenPrompt.

3.1 Combinability

In the NLP world, we usually adopt different PLMs with corresponding objective functions to different underlying tasks (roughly, classification and generation). But in prompt learning, given that the core idea of the framework is to mimic pre-training tasks in the downstream task, which are essentially "predicting words based on context", we can further unify the execution of downstream tasks. OpenPrompt supports a combination of tasks (classification and generation), PLMs (MLM, LM and Seq2Seq), and prompt modules (different templates and verbalizers) in a flexible way. For example, from a model perspective, T5 (Raffel et al., 2019) is not only used for span prediction and GPT (Brown et al., 2020) is not only used for generative tasks. From the perspective of prompting, prefix-tuning can also be used for classification, and soft prompt can be used for generation. All these combinations can easily be implemented and validated on NLP tasks in our framework so that we can better understand the mechanisms involved.

3.2 Pre-trained Language Models

One core idea of prompt-learning is to use additional context with masked tokens to imitate the pre-training objectives of PLMs and better stimulate these models. Hence, the choice of PLMs is crucial to the whole pipeline of prompt-learning. PLMs could be roughly divided into three groups according to their pre-training objectives.

The first group of PLMs use masked language modeling (MLM) to reconstruct a sequence corrupted by random masked tokens, where only the losses of the masked tokens are computed. Typical PLMs with MLM objective include BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), etc, and such an objective is regarded suitable for natural language understanding (NLU). The second group exploits the autoregressive-style language modeling (LM) to predict the current token according to its leading tokens. GPT-3 (Brown et al., 2020) is one of the representative works adopting this objective. The third part is the sequence-to-sequence (Seq2Seq) models, which aim to generate a sequence with a decoder conditioned on a separate encoder for an input sequence. Typical seq2seq PLMs

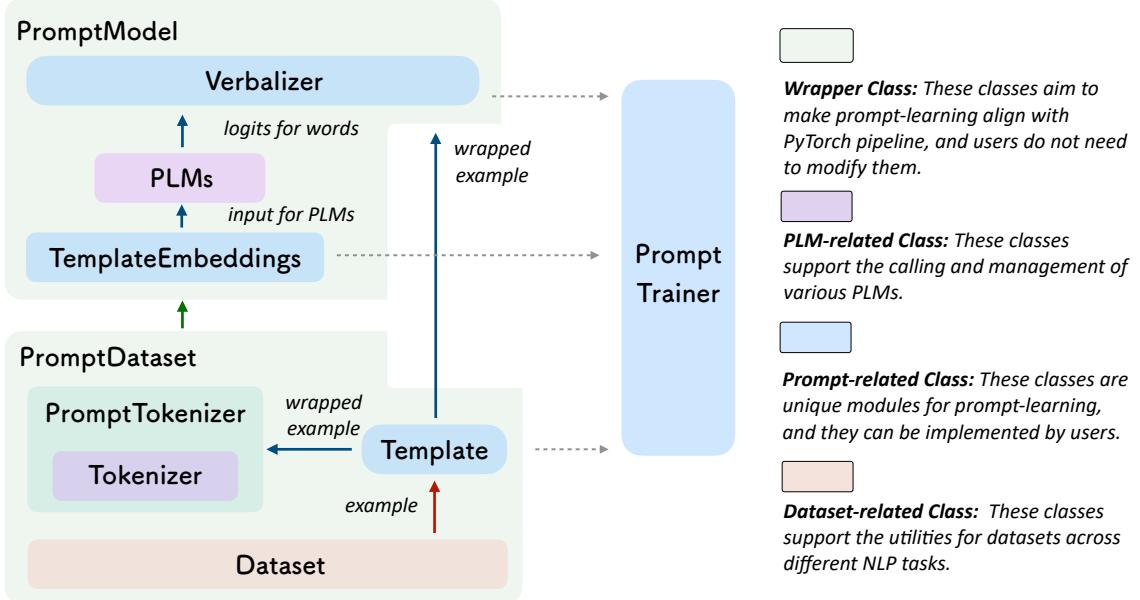


Figure 1: The overall architecture of OpenPrompt. Note that according to the prompt-learning strategies, not all the modules are necessarily used. For example, in generation tasks, there are no verbalizers in the learning procedure. The `PromptTrainer` is a controller that controls the data flow and the training process with some unique attributes, users can also implement the training process in a conventional fashion.

include T5 (Raffel et al., 2020), MASS (Song et al., 2019) and BART (Lewis et al., 2020), etc.

Different PLMs have different attributes, resulting in various adaptation capabilities for different NLP tasks in prompt-learning. Practically in OpenPrompt, we support directly loading PLMs from huggingface transformers¹ (Wolf et al., 2020), and PLMs implemented by other libraries will be supported in the future. Once the PLM is determined, researchers could deploy a known valid prompt-learning pipeline (e.g., RoBERTa for few-shot sentiment classification) or explore other uses of PLM that could exploit its potential. Users of OpenPrompt do not need to implement objective heads for different PLMs to calculate the corresponding loss, a unified interface can perform these operations automatically (§ 3.6).

3.3 Tokenization

Tokenization is a crucial step in processing data for NLP, and it faces new challenges in prompt-learning. After designing the template, the specific implementation of the tokenization for original input and the designed template could be time-consuming and error-prone. First, in prompt-learning, some specific information such as the indices of entities and masked tokens should be

carefully tackled in tokenization. Some small errors, such as the mismatch of masked token indices, may lead to serious consequences. Moreover, concatenation and truncation issues after tokenization (templates are not supposed to be truncated) should also be handled. Since different PLMs may have different tokenization strategies, we should also consider the inconsistency in the details of additional context processing.

In OpenPrompt, we specifically design the tokenization module for prompt-learning and significantly simplify the process. By using our encapsulated data processing APIs, users could use the human-readable style to design templates and conveniently operate on the input and the template at the same time. Our component integrates complex information from input and template and then conducts tokenization. Based on the choice of PLMs (MLM, LM, and Seq2Seq), OpenPrompt automatically chooses the appropriate tokenizer in prompt-learning, which could save considerable time for users to process prompt-related data.

3.4 Templates

As one of the central parts of prompt-learning, a template module wraps the original text with the textual or soft-encoding template. A template normally contains contextual tokens (textual or soft)

¹<https://huggingface.co/models>

```

1 # Example A. Hard prompt for topic classification
2 a {"mask"} news: {"meta": "title"} {"meta": "description"}
3
4 # Example B. Hard prompt for entity typing
5 {"meta": "sentence"}. In this sentence, {"meta": "entity"} is a {"mask"}, ,
6
7 # Example C. Soft prompt (initialized by textual tokens)
8 {"meta": "premise"} {"meta": "hypothesis"} {"soft": "Does the first sentence entails
      the second ?"} {"mask"} {"soft"}.
9
10 # Example D. The power of scale
11 {"soft": None, "duplicate": 100} {"meta": "text"} {"mask"}
12
13 # Example E. Post processing script support
14 # e.g. write an lambda expression to strip the final punctuation in data
15 {"meta": "context", "post_processing": lambda s: s.rstrip(string.punctuation)}. {""
      soft": "It was"} {"mask"}
16
17 # Example F. Mixed prompt with two shared soft tokens
18 {"meta": "premise"} {"meta": "hypothesis"} {"soft": "Does"} {"soft": "the", "soft_id
      ": 1} first sentence entails {"soft_id": 1} second?
19
20 # Example G. Specify the title should not be truncated
21 a {"mask"} news: {"meta": "title", "shortenable": False} {"meta": "description"}

```

Figure 2: Some examples of our template language. In our template language, we can use the key “meta” to refer the original input text (Example B), parts of the original input (Example A, C, G), or other key information. We can also freely specify which tokens are hard and which are soft (and their initialization strategy). We could assign an id for a soft token to specify which tokens are sharing embeddings (Example F). OpenPrompt also supports the post processing (Example E) for each token, e.g., lambda expression or MLP.

and masked tokens. In OpenPrompt, all the templates are inherited from a common base class with universal attributes and abstract methods.

Previous works design a wide variety of templates, including manually written template (Schick and Schütze, 2021) and pure soft template (Lester et al., 2021). Gu et al. (2021) report a mix of manual template tokens and soft (trainable) tokens sometimes yields better results than separate manual template and soft template. In Liu et al. (2021b), a promising performance is achieved by fixing the majority of manual tokens while tuning a small number of the others. In Han et al. (2021b), the template is contextualized, which needs to be filled with the head entity and the tail entity to form a complete one, moreover, the output of multiple positions is used in the loss calculation in their template. Logan IV et al. (2021) design null template with simple concatenation of the inputs and an appended <mask> token.

It’s not reasonable to design a template format for each prompt since it will require high learning cost for practical use. To this end, in OpenPrompt, we design a template language to ease the problem, with which we can construct various types of templates under a unified paradigm. Our template

language takes insight from the dict grammar of Python. And such a design ensures flexibility and clarity at the same time, allowing users to build different prompts with relative ease.

More specifically, a template node is a text (or empty text) with an attributes’ description. In our template language, one is free to edit the attributes of each token in the template, such as which characters are shared embedding, how the characters are post-processed (e.g. by MLP), etc. We show some template examples in Figure 2, and the detailed tutorial for writing templates is in our documentation <https://thunlp.github.io/OpenPrompt>.

```

1 from openprompt import
      ManualVerbalizer
2
3 promptVerbalizer = ManualVerbalizer(
4     classes = classes,
5     label_words = {
6         "negative": ["bad"],
7         "positive": ["good", "
      wonderful", "great"],
8     },
9     tokenizer = bertTokenizer,
10 )

```

Figure 3: An example to define a Verbalizer, the number of the label words for each class is flexible.

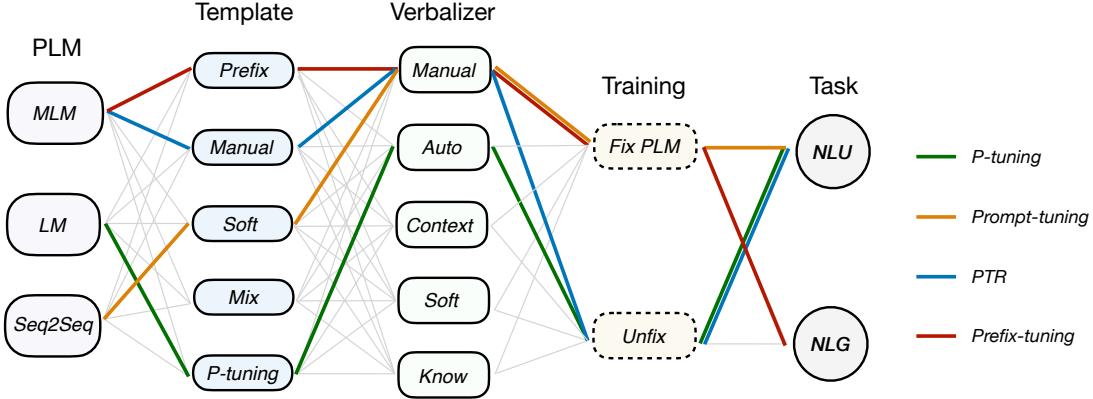


Figure 4: The illustration of the validation space of OpenPrompt. By driving different modules of the framework, we could implement and evaluate different methods on a broad set of NLP tasks. We show four examples in this illustration, the colored lines denote the implementation flow of the corresponding method.

3.5 Verbalizers

When it comes to prompt-based classification, a verbalizer class should be constructed to map original labels to label words in the vocabulary. When a PLM predicts a probability distribution over the vocabulary for one masked position, a verbalizer will extract the logits of label words and integrate the logits of label words to the corresponding class, thereby responsible for the loss calculation. Figure 3 shows a simple way to define a binary sentiment classification verbalizer.

Similar to templates, all the verbalizer classes are also inherited from a common base class with necessary attributes and abstract methods. Additional to manually-defined verbalizers, we implement automatic verbalizers like AutomaticVerbalizer and KnowledgeableVerbalizer (Hu et al., 2021). Moreover, important operations like calibrations (Zhao et al., 2021) are also realized in OpenPrompt.

3.6 PromptModel

In OpenPrompt, we use a `PromptModel` object to be responsible for training and inference, which contains a PLM, a `Template` object, and a `Verbalizer` object (optional). Users could flexibly combine these modules and define advanced interactions among them. A model-agnostic forward method is implemented in the base class to predict words for the masked positions. One goal of this module is that users do not need to specifically implement heads for different PLMs, but use a unified API to “predict words for positions that need to be predicted” regardless of the pre-training objective. An example to define a `PromptModel` is shown in Figure 5.

```

1 from openprompt import
2     PromptForClassification
3
4 promptModel = PromptForClassification(
5     template = promptTemplate,
6     model = bertModel,
7     verbalizer = promptVerbalizer,
8 )
9
10 promptModel.eval()
11 with torch.no_grad():
12     for batch in data_loader:
13         logits = promptModel(batch)
14         preds = torch.argmax(logits,
15             dim = -1)
16         print(classes[preds])

```

Figure 5: An example to define a `PromptModel` and conduct evaluation.

3.7 Training

From the perspective of trainable parameters, the training of prompt-learning could be divided into two types of strategies. The first strategy simultaneously tunes the prompts and the PLM, which is verified to be effective in a low-data regime (OpenPrompt also provides a `FewshotSampler` to support the few-shot learning scenario). The second strategy is to only train the parameters of prompts and keep the PLM frozen, this is regarded as a parameter-efficient tuning method and is considered as a promising way to stimulate super-large PLMs. Both of these strategies can be called with one click in the trainer (or runner) module of OpenPrompt. Trainer modules in OpenPrompt implement training process accompanied with prompt-oriented training tricks, e.g. the ensemble of templates. Meanwhile, OpenPrompt supports experimentation through configuration to easily drive large-scale empirical study.

4 Evaluation

OpenPrompt aims to support a broad set of NLP tasks under the paradigm of prompt-learning. In terms of evaluation, we use OpenPrompt to implement various baselines and assess them on the corresponding NLP tasks. We show the validation space in Figure 4. And the evaluation tasks include WebNLG (Gardent et al., 2017) for conditional generation, GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) for natural language understanding; SemEval (Hendrickx et al., 2010) for relation extraction; Few-NERD (Ding et al., 2021b) for fine-grained entity typing; MNLI (Williams et al., 2017), AG’s News (Zhang et al., 2015), DBpedia (Lehmann et al., 2015) and IMDB (Maas et al., 2011) for text classification; LAMA (Petroni et al., 2019) for knowledge probing. The processors of these datasets have already been implemented in OpenPrompt, and they are all inherited from a common base `DataProcessor` class. To keep the results up to date, we are constantly updating and reporting the latest results on our GitHub repository <https://github.com/thunlp/OpenPrompt>.

5 Conclusion and Future Work

We propose OpenPrompt, a unified, easy-to-use and extensible toolkit for prompt-learning. OpenPrompt establishes a unified framework with clearly defined blocks and flexible interactions to support solid research on prompt-learning. At the application level, OpenPrompt could facilitate researchers and developers to effectively and efficiently deploy prompt-learning pipelines. In the future, we will continue to integrate new techniques and features to OpenPrompt to facilitate the research progress of prompt-learning.

References

- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. **Language models are few-shot learners.** *arXiv preprint arXiv:2005.14165*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of deep bidirectional transformers for language understanding.** In *Proceedings of ACL*, pages 4171–4186, Minneapolis, Minnesota.
- Ning Ding, Yulin Chen, Xu Han, Guangwei Xu, Pengjun Xie, Hai-Tao Zheng, Zhiyuan Liu, Juanzi Li, and Hong-Gee Kim. 2021a. **Prompt-learning for fine-grained entity typing.** *Arxiv preprint, 2108.10604*.
- Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Hai-Tao Zheng, and Zhiyuan Liu. 2021b. **Few-nerd: A few-shot named entity recognition dataset.** In *Proceedings of ACL*.
- Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. **Making pre-trained language models better few-shot learners.** In *Proceedings of ACL*, pages 3816–3830, Online.
- Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. **The webnlg challenge: Generating text from rdf data.** In *Proceedings of INLG*, pages 124–133.
- Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2021. **Ppt: Pre-trained prompt tuning for few-shot learning.** *arXiv preprint arXiv:2109.04332*.
- Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. 2021a. **Pre-trained models: Past, present and future.** *ArXiv preprint, abs/2106.07139*.
- Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021b. **Ptr: Prompt tuning with rules for text classification.** *ArXiv preprint, 2105.11259*.
- Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. **SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals.** In *Proceedings of SemEval*, pages 33–38.
- Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Juanzi Li, and Maosong Sun. 2021. **Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification.** *ArXiv preprint, 2108.02035*.
- Teven Le Scao and Alexander M Rush. 2021. **How many data points is a prompt worth?** In *Proceedings of NAACL*, pages 2627–2636.
- Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. **Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia.** *Semantic web, 6(2):167–195*.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. **The power of scale for parameter-efficient prompt tuning.** *ArXiv preprint, abs/2104.08691*.

- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. **BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension**. In *Proceedings of ACL*, pages 7871–7880, Online.
- Xiang Lisa Li and Percy Liang. 2021. **Prefix-tuning: Optimizing continuous prompts for generation**. In *Proceedings ACL*, pages 4582–4597, Online. Association for Computational Linguistics.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. **Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing**. *ArXiv preprint*, abs/2107.13586.
- Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. **Gpt understands, too**. *arXiv preprint arXiv:2103.10385*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. **RoBERTa: A robustly optimized BERT pretraining approach**. *ArXiv preprint*, abs/1907.11692.
- Robert L Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2021. **Cutting down on prompts and parameters: Simple few-shot learning with language models**. *arXiv preprint arXiv:2106.13353*.
- Andrew 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 ACL*.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. **Pytorch: An imperative style, high-performance deep learning library**. *Proceedings of NeurIPS*, 32:8026–8037.
- Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. **Language models as knowledge bases?** *arXiv preprint arXiv:1909.01066*.
- Yujia Qin, Xiaozhi Wang, Yusheng Su, Yankai Lin, Ning Ding, Zhiyuan Liu, Juanzi Li, Lei Hou, Peng Li, Maosong Sun, et al. 2021. **Exploring low-dimensional intrinsic task subspace via prompt tuning**. *arXiv preprint arXiv:2110.07867*.
- Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. **Pre-trained models for natural language processing: A survey**. *Science China Technological Sciences*, pages 1–26.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. **Exploring the limits of transfer learning with a unified text-to-text transformer**. *ArXiv preprint*, abs/1910.10683.
- 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**. *Journal of Machine Learning Research*, 21(140):1–67.
- Timo Schick and Hinrich Schütze. 2021. **Exploiting cloze-questions for few-shot text classification and natural language inference**. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.
- Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. **Mass: Masked sequence to sequence pre-training for language generation**. *arXiv preprint arXiv:1905.02450*.
- Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou, and Daniel Cer. 2021. **Spot: Better frozen model adaptation through soft prompt transfer**. *arXiv preprint arXiv:2110.07904*.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. **Super glue: A stickier benchmark for general-purpose language understanding systems**. *arXiv preprint arXiv:1905.00537*.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. **Glue: A multi-task benchmark and analysis platform for natural language understanding**. *arXiv preprint arXiv:1804.07461*.
- Colin Wei, Sang Michael Xie, and Tengyu Ma. 2021. **Why do pretrained language models help in downstream tasks? an analysis of head and prompt tuning**.
- Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. **A broad-coverage challenge corpus for sentence understanding through inference**. *arXiv preprint arXiv:1704.05426*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrette Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. **Transformers: State-of-the-art natural language processing**. In *Proceedings of EMNLP*, pages 38–45, Online.
- Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. **Character-level convolutional networks for text classification**. *Advances in neural information processing systems*.

Tony Z Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *arXiv preprint arXiv:2102.09690*.