HugNLP: A Unified and Comprehensive Library for Natural Language Processing

Jianing Wang
East China Normal University
Shanghai, China
lygwjn@gmail.com

Nuo Chen
East China Normal University
Shanghai, China
nuochen@stu.ecnu.edu.cn

Qiushi Sun
National University of Singapore
Singapore
qiushisun@u.nus.edu

Wenkang Huang
Ant Group
Shanghai, China
wenkang.hwk@alibaba-inc.com

Chengyu Wang
Alibaba Group
Hangzhou, China
chengyu.wcy@alibaba-inc.com

Ming Gao
East China Normal University
Shanghai, China
mgao@dase.ecnu.edu.cn

ABSTRACT

In this paper, we introduce HugNLP, a unified and comprehensive library for natural language processing (NLP) with the prevalent backend of Hugging Face Transformers, which is designed for NLP researchers to easily utilize off-the-shelf algorithms and develop novel methods with user-defined models and tasks in real-world scenarios. HugNLP consists of a hierarchical structure including models, processors and applications that unifies the learning process of pre-trained language models (PLMs) on different NLP tasks. Additionally, we present some featured NLP applications to show the effectiveness of HugNLP, such as knowledge-enhanced PLMs, universal information extraction, low-resource mining, and code understanding and generation, etc. The source code will be released on GitHub (https://github.com/HugAILab/HugNLP).

CCS CONCEPTS

• **Computing methodologies → Natural language processing**.

KEYWORDS

Natural Language Processing, Pre-trained Language Models, Deep Learning Framework

ACM Reference Format:

Jianing Wang, Nuo Chen, Qiushi Sun, Wenkang Huang, Chengyu Wang, and Ming Gao. 2023. HugNLP: A Unified and Comprehensive Library for Natural Language Processing. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3583780.3614742

1 INTRODUCTION

Recently, pre-trained language models (PLMs) have become the imperative infrastructure in natural language processing (NLP) tasks [5, 18, 43], which bring substantial improvements by a two-stage training strategy: *pre-train* and *fine-tune*. Benefiting from this strategy, a branch of methods arises to improve the models' effectiveness, promoting NLP's development in both academia and industry [12, 16].

Yet, many existing approaches follow different patterns and code architectures, it is not easy to obtain high-performing models and develop them easily for researchers. To fill this gap, this paper presents a unified and comprehensive open-source library to allow researchers to develop and evaluate NLP models more efficiently and effectively. We mainly utilize Hugging Face Transformers[1] as the prevalent backend, which provides abundant backbones of different scale-sizes of PLMs. Thanks to Hugging Face, we name our framework as HugNLP to fully extend Hugging Face Transformers into an NLP-style library. HugNLP consists of some well-designed components, such as *Models*, *Processors*, and *Applications*. Concretely, 1) for *Models*, we provide some popular PLMs, including BERT [5], RoBERTa [18], DeBERTa [9], GPT-2 [25] and T5 [26], etc. Based on these PLMs, we develop task-specific modules for pre-training (e.g., masked language modeling (MLM), casual language modeling (CLM)) and fine-tuning (e.g., sequence classifying and matching, span extraction, text generation). We also provide some prompt-based techniques for PLMs, including PET [27], P-tuning [17], Prefix-tuning [15], Adapter-tuning [10], In-context learning [6] and Chain-of-Thought prompting [40]. 2) In *Processors*, we develop relevant data processing tools[2] for some commonly used benchmark datasets and business-specific corpora. 3) In *Applications*, we present core capacities to support the upper-layer components. Specifically, our proposed KP-PLM [33] enables plug-and-play knowledge injection in model pre-training and fine-tuning via converting structure knowledge into unified language prompts. We also develop some products: 1) HugIE: a unified information extraction framework, and 2)HugChat[3]: a ChatGPT-like training pipeline for large language models (LLMs). HugNLP also integrates some novel algorithms and applications, such as uncertainty-aware self-training [21, 34], code understanding and generation [7, 36, 38].

Overall, HugNLP has the following features.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom.

ACM ISBN 979-8-4007-0124-5/23/10…$15.00
https://doi.org/10.1145/3583780.3614742

[1]https://HuggingFace.co/.
[2]The *Processor* is related to the task format. For example, we tailor some benchmark datasets, such as Chinese CLUE [42], GLUE [31], etc.
[3]HugChat: Small ChatGPT-like Models via Generative Instruction-tuning

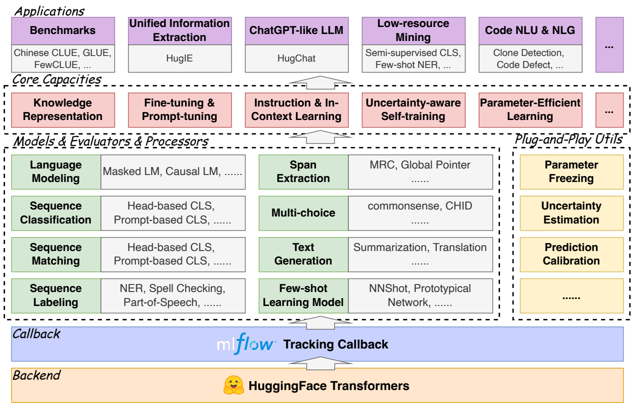

Figure 1: An overview of the HugNLP library.

- HugNLP offers a range of pre-built components and modules (i.e., *Models*, *Processors*, *Applications*) that can be used to speed up the development process and simplify the implementation of complex NLP models and tasks.
- HugNLP can also be easily integrated into existing workflows and customized to meet the specific needs of individual researchers or projects, ensuring the framework's scalability and flexibility.
- HugNLP is equipped with some novel core capacities, such as knowledge-enhanced pre-training, prompt-based fine-tuning, instruction and in-context learning, uncertainty-aware self-training, and parameter-efficient learning. We thus develop some featured products or solutions on real-world application scenarios, e.g., HugIE, and HugChat.
- HugNLP is based on PyTorch and Hugging Face, which are widely used tools and platforms in the NLP community, allowing researchers to leverage their strengths and apply them to both academics and industry scenarios [13, 24, 32, 41].

2 HUGNLP

2.1 Overview

HugNLP is an open-sourced library with a hierarchical structure. As shown in Figure 1. The backend is the prevalent Hugging Face Transformers platform that provides multiple transformer-based models and task trainers. In other words, HugNLP can be seen as a customized NLP platform for efficient training and evaluation. In addition, HugNLP integrates *MLFlow*, which is a novel tracking callback toolkit for model training and experiment result analysis. Users can simply add configure parameters `tracking_uri` in the training script, and observe the tracking records after running *MLFlow* server.

HugNLP consists of three key components, including *Models*, *Processors*, and *Applications*. Users can directly select the pre-built settings for some common tasks, or develop special user-defined training solutions in real-world application scenarios. We will provide a detailed description in the following sections.

```
from tools.model_utils.parameter_freeze import
    ParameterFreeze
freezer = ParameterFreeze()
class BertForSequenceClassification(BertPreTrained
    Model):
    def __init__(self, config):
        super().__init__(config)
        self.num_labels = config.num_labels
        self.config = config
        self.bert = BertModel(config)
        # freeze the backbone
        if self.config.use_freezing:
            self.bert = freezer.freeze_lm(self.bert)
        self.classifier = torch.nn.Linear(
            config.hidden_size, config.num_labels)
        self.init_weights()
```

Code 1: A model case of parameter freezing.

2.2 Library Architecture

Models. In *Models*, we provide some popular transformer-based models as backbones, such as BERT, RoBERTa, GPT-2, etc. We also release our pre-built KP-PLM, a novel knowledge-enhanced pre-training model which leverages *knowledge prompting* [33] paradigm to inject factual knowledge and can be easily used for arbitrary PLMs. Apart from basic PLMs, we also implement some task-specific models, involving sequence classification, matching, labeling, span extraction, multi-choice, and text generation. Particularly, we develop standard fine-tuning (based on CLS Head [4]) and prompt-tuning models [5] that enable PLM tuning on classification tasks. For few-shot learning settings, HugNLP provides a prototypical network [28] in both few-shot text classification and named entity recognition (NER).

In addition, we also incorporate some *plug-and-play utils* in HugNLP. 1) *Parameter Freezing.* If we want to perform parameter-efficient learning [20], which aims to freeze some parameters in PLMs to improve the training efficiency, we can set the configure `use_freezing` and freeze the backbone. A use case is shown in Code 1. 2) *Uncertainty Estimation* aims to calculate the model certainty when in semi-supervised learning [21]. 3) We also design *Prediction Calibration*, which can be used to further improve the accuracy by calibrating the distribution and alleviating the semantics bias problem [46].

Processors. HugNLP aims to load the dataset and process the task examples in a pipeline, containing sentence tokenization, sampling, and tensor generation. Specifically, users can directly obtain the data through `load_dataset`, which can directly download it from the Internet or load it from the local disk. For different tasks, users should define a task-specific data collator, which aims to transform the original examples into model input tensor features.

Applications. It provides rich modules for users to build real-world applications and products by selecting among an array of settings from *Models* and *Processors*. More details are shown in Section 2.4.

[4]For standard fine-tuning, we need to add a classification head (CLS head) on the PLM and obtain the probability distribution of each class. The parameters of the CLS head are randomly initialized.

[5]Different from fine-tuning, prompt-tuning can reuse the pre-training objective (e.g., MLM, CLM) to perform classifying on the masked token. It requires a task-orient template (e.g., "It was [MASK].") and the label word mapping (e.g., "great" maps to "positive" class in sentiment analysis task.)

```
python3 hugnlp_runner.py \
  --model_name_or_path=$path \
  --data_dir=$data_path \
  --output_dir=./outputs/glue/$glue_task \
  --seed=42 \
  --max_seq_length=$len \
  --do_train \
  --do_eval \
  --gradient_accumulation_steps=1 \
  --evaluation_strategy=steps \
  --learning_rate=1e-5 \
  --num_train_epochs=10 \
  --task_name=clue \
  --task_type=head_cls \
  --model_type=bert \
  --user_defined="data_name=rte" \
```

Code 2: An application case of sequence classification for GLUE benchmark.

2.3 Core Capacities

To further improve the effectiveness of HugNLP, we design multiple core capacities in the following.

Knowledge-enhanced Pre-training. Conventional pre-training methods lack factual knowledge [22, 45]. To deal with this issue, we present KP-PLM [33] with a novel knowledge prompting paradigm for knowledge-enhanced pre-training. Specifically, we construct a knowledge sub-graph for each input text by recognizing entities and aligning with the knowledge base (e.g., Wikidata5M [6]) and decompose this sub-graph into multiple relation paths, which can be directly transformed into language prompts. KP-PLM can be easily applied to other PLMs without introducing extra parameters as knowledge encoders.

Prompt-based Fine-tuning. Prompt-based fine-tuning aims to reuse the pre-training objective (e.g., MLM) and utilizes a well-designed template and verbalizer to make predictions, which has achieved great success in low-resource settings. We integrate some novel approaches into HugNLP, such as PET [27], P-tuning [17], etc. We also build some parameter-efficient learning to make it more effective when training LLMs.

Instruction-tuning and In-Context Learning. Instruction-tuning [39] and in-context learning [2] enable few/zero-shot learning without parameter update, which aims to concatenate the task-aware instructions or example-based demonstrations to prompt GPT-style causal language models to generate reliable responses. These approaches are mainly used in recent LLMs, such as ChatGPT[7], LLaMA, and LangChain[8]. So, all the NLP tasks can be unified into the same format and can substantially improve the models' generalization. Inspired by this idea, we also extend it into two other paradigms: 1) extractive-style paradigm: we unify various NLP tasks into span extraction, which is the same as extractive question answering [14], and 2) inference-style paradigm: all the tasks can be viewed as natural language inference to match the relations between inputs and outputs [35].

Uncertainty-aware Self-training. Self-training can address the labeled data scarcity issue by leveraging the large-scale unlabeled data in addition to labeled data, which is one of the mature paradigms in semi-supervised learning [1, 3, 23]. However, the standard self-training may generate too many noises, inevitably degrading the model performance due to the confirmation bias. Thus, we present uncertainty-aware self-training. Specifically, we train a teacher model on few-shot labeled data, and then use Monte Carlo (MC) dropout technique in Bayesian neural network (BNN) [8] to approximate the model certainty, and judiciously select the examples that have a higher model certainty of the teacher.

Parameter-efficient Learning. To improve the training efficiency of HugNLP, we also implement parameter-efficient learning, which aims to freeze some parameters in the backbone so that we only tune a few parameters during model training. We develop some novel parameter-efficient learning approaches, such as Prefix-tuning [15], Adapter-tuning [10], BitFit [44] and LoRA [11], etc.

```
>>> from applications.information_extraction.HugIE.api_test import HugIEAPI
>>> model_type = 'bert'
>>> hugie_model_name_or_path = 'wjn1996/wjn1996-hugnlp-hugie-large-zh'
>>> hugie = HugIEAPI(model_type, hugie_model_name_or_path)
>>> text = '北京在2008年和2022年分别举办了夏季奥运会和冬季奥运会'
>>> # Beijing has posted the Summer and Winter Olympics in 2008 and 2022, re
spectively.
>>> entity = '2008年奥运会' # 2008 Olympics Games
>>> relation = '举办地' # host place
>>> predictions, _ = hugie.request(text, entity, relation)
>>> print(predictions)
{0: ['北京']}
>>> # {0: ['Beijing']}
```

Figure 2: An application case of HugIE.

PLMs	AFQMC	CMNLI	CSL	IFLYTEK	OCNLI	TNEWS	WSC	Avg.
BERT-base	72.30	75.91	80.83	60.11	78.52	57.18	75.89	72.04
BERT-large	72.91	77.62	81.30	60.77	78.71	57.77	78.28	72.60
RoBERTa-base	73.33	81.05	80.17	60.81	80.88	57.69	86.74	74.10
RoBERTa-large	74.66	80.50	82.60	61.37	82.19	58.54	87.53	75.33
MacBERT-base	74.23	80.65	81.63	61.14	80.65	57.65	80.26	73.80
MacBERT-large	74.66	81.19	83.70	62.05	81.92	59.03	86.74	75.46

Table 1: Accuracy (%) of different tasks in the CLUE benchmark.

2.4 Featured Applications

Benchmark Tuning. We develop the training application for some popular benchmarks, such as Chinese CLUE and GLUE. We use both standard fine-tuning and prompt-based fine-tuning paradigms to tune PLMs over these benchmarks. The case of this application is shown in Code 2.

Universal Information Extraction based on Extractive Instruction. We develop HugIE, a novel universal information extraction toolkit based on HugNLP. Specifically, we collect multiple Chinese NER and event extraction datasets from ModelScope [9] and QianYan [10]. Then, we use the core capacity of extractive-style instruction with a global pointer [29] to pre-train a universal information extraction model. We also upload the trained model to Hugging Face [11]. An example of using HugIE is shown in Figure 2.

Low-resource Tuning for PLMs. For low-resource settings, we have integrated two core capacities of prompt-tuning and uncertainty-aware self-training to further improve the performance with limited labeled data. In other words, prompt-tuning can fully reuse the

[6]https://deepgraphlearning.github.io/project/wikidata5m.
[7]https://chat.openai.com/.
[8]https://langchain.com/.
[9]https://modelscope.cn/datasets
[10]https://www.luge.ai
[11]https://HuggingFace.co/wjn1996/wjn1996-hugnlp-hugie-large-zh.

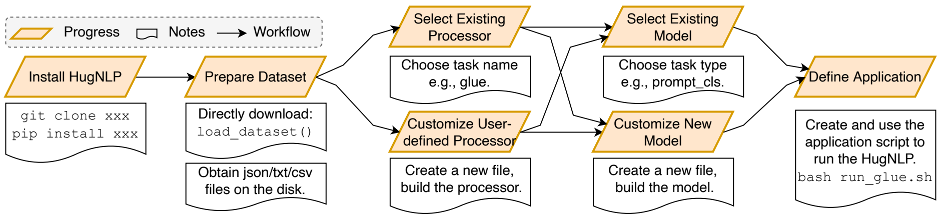

Figure 3: The development workflow of HugNLP.

Paradigms	Methods	SST-2 (acc)	SST-5 (acc)	MR (acc)	CR (acc)	MPQA (acc)	Subj (acc)	TREC (acc)	CoLA (matt.)	Avg.
PT-Zero	RoBERTa	82.57	29.46	**65.10**	**82.15**	49.90	**69.20**	20.80	-4.89	49.29
	KP-PLM	**84.15**	**30.67**	64.15	81.60	**53.80**	68.70	**24.80**	**-2.99**	**50.61**
PT-Few	RoBERTa	86.35±1.3	36.79±2.0	**83.35**±0.9	**88.85**±1.4	66.40±1.9	89.25±2.6	76.80±5.0	6.61±6.9	66.80
	KP-PLM	**90.71**±1.0	**44.21**±2.9	82.00±1.5	85.35±0.4	**67.30**±1.2	**91.45**±0.4	**81.00**±3.3	**24.28**±11.3	**70.79**
FT-Full	RoBERTa	94.90	56.90	**89.60**	88.80	86.30	**96.50**	**97.10**	63.90	84.25
	KP-PLM	**95.30**	**57.63**	89.20	**89.10**	**87.40**	96.20	**97.10**	**64.87**	**84.60**

Table 2: The comparison between KP-PLM and RoBERTa-base over multiple natural language understanding (NLU) tasks in terms of acc/f1/matt. (%) and standard deviation with three paradigms, such as zero-shot prompt-tuning (PT-Zero), few-shot prompt-tuning (PT-Few), and full-data fine-tuning (FT-Full).

prior knowledge derived from PLMs to achieve high grades with few examples, while self-training can augment unlabeled data to enhance effectiveness.

Code Understanding and Generation. In addition to traditional NLP tasks, we also consider the scenario of code understanding and generation, such as clone detection, defect detection, and code summarization [19].

2.5 Development Workflow

HugNLP is easy to use and develop. We draw a workflow in Figure 3 to show how to develop a new running task. It consists of five main steps, including library installation, data preparation, processor selection or design, model selection or design, and application design. This illustrates that HugNLP can simplify the implementation of complex NLP models and tasks.

3 EXPERIMENTAL PERFORMANCES

In this section, we empirically examine the effectiveness and efficiency of the HugNLP toolkit on some public datasets.

3.1 Performance of Benchmarks

To validate the effectiveness of HugNLP on both fine-tuning and prompt-tuning, we choose Chinese CLUE [42] and GLUE benchmarks [31]. For Chinese CLUE, we choose different sizes of BERT, RoBERTa and MacBERT [4] and report the accuracy over the development sets of each task in Tables 1. For GLUE, we perform full-resource fine-tuning (FT-full), few-shot prompt-tuning (PT-few), and zero-shot prompt-tuning (PT-zero) based on our proposed KP-PLM. We select RoBERTa as the strong baseline and report the accuracy results with standard deviation in Table 2. The obtained comparable performance has shown the reliability of HugNLP in

Methods	RTE	CB	AGNews	Avg.
Few Labeled Data (16-shot)				
Fine-Tuning	54.4±3.9	74.5±2.6	88.9±2.7	72.60
Few Labeled Data (16-shot) + Unlabeled Data				
UST	55.6±2.6	76.0±3.1	89.3±3.5	73.63
CEST	57.0±1.9	78.1±2.7	88.5±2.2	74.53
LiST	**60.8±2.5**	**79.7±2.9**	**90.3±2.5**	**76.93**

Table 3: Accuracy (%) of uncertain-aware self-training with only 16 labeled examples per class.

both full and low-resource scenarios, which achieves similar performance compared to other open-source frameworks and their original implementations [32].

3.2 Effectiveness of Self-training

We end this section with an additional validation on the self-training. We choose some recent methods (using uncertainty estimation) to evaluate the implementations of HugNLP, including UST [21], CEST [30], and LiST [37]. Results in Table 3 show that self-training can make substantial improvements in low-resource scenarios.

4 CONCLUSION

In this paper, we introduce HugNLP, a unified and comprehensive library based on PyTorch and Hugging Face, allowing researchers to apply it to different academics and industry scenarios. HugNLP consists of three key components (i.e., *Processors*, *Models* and *Applications*) and multiple pre-built core capacities and plug-and-play utils. Finally, we perform some evaluation of different aspects of applications, and the results demonstrate its efficiency and effectiveness. We think HugNLP can promote research and development for NLP applications.

REFERENCES

[1] Massih-Reza Amini, Vasilii Feofanov, Loïc Pauletto, Emilie Devijver, and Yury Maximov. 2022. Self-Training: A Survey. *CoRR* abs/2202.12040 (2022).

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeurIPS*.

[3] Nitesh V. Chawla and Grigoris I. Karakoulas. 2005. Learning From Labeled And Unlabeled Data: An Empirical Study Across Techniques And Domains. *JAIS* 23 (2005), 331–366.

[4] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting Pre-Trained Models for Chinese Natural Language Processing. In *EMNLP (Findings)*. 657–668.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. 4171–4186.

[6] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A Survey for In-context Learning. *CoRR* abs/2301.00234 (2023). https://doi.org/10.48550/arXiv.2301.00234 arXiv:2301.00234

[7] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *EMNLP*. Association for Computational Linguistics, Online, 1536–1547.

[8] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *ICML*, Vol. 48. 1050–1059.

[9] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-Enhanced Bert with Disentangled Attention. In *ICLR*. OpenReview.net.

[10] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *ICML (Proceedings of Machine Learning Research, Vol. 97)*. 2790–2799.

[11] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

[12] Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. 2022. A Survey of Knowledge-Enhanced Pre-trained Language Models. *CoRR* abs/2211.05994 (2022).

[13] Shengding Hu, Ning Ding, Weilin Zhao, Xingtai Lv, Zhen Zhang, Zhiyuan Liu, and Maosong Sun. 2023. OpenDelta: A Plug-and-play Library for Parameter-efficient Adaptation of Pre-trained Models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Association for Computational Linguistics, Toronto, Canada, 274–281.

[14] Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Unifying Question Answering and Text Classification via Span Extraction. *CoRR* abs/1904.09286 (2019).

[15] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *ACL*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, 4582–4597.

[16] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* 55, 9 (2023), 195:1–195:35.

[17] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT Understands, Too. *CoRR* abs/2103.10385 (2021).

[18] 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. *CoRR* abs/1907.11692 (2019).

[19] Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin B. Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie Liu. 2021. CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation. *CoRR* abs/2102.04664 (2021).

[20] Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Scott Yih, and Madian Khabsa. 2022. UniPELT: A Unified Framework for Parameter-Efficient Language Model Tuning. In *ACL*. Association for Computational Linguistics, 6253–6264.

[21] Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. Uncertainty-aware Self-training for Few-shot Text Classification. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

[22] Xiaoman Pan, Wenlin Yao, Hongming Zhang, Dian Yu, Dong Yu, and Jianshu Chen. 2022. Knowledge-in-Context: Towards Knowledgeable Semi-Parametric Language Models. *CoRR* abs/2210.16433 (2022).

[23] Guo-Jun Qi and Jiebo Luo. 2022. Small Data Challenges in Big Data Era: A Survey of Recent Progress on Unsupervised and Semi-Supervised Methods. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 4 (2022), 2168–2187.

[24] Minghui Qiu, Peng Li, Chengyu Wang, Haojie Pan, Ang Wang, Cen Chen, Xianyan Jia, Yaliang Li, Jun Huang, Deng Cai, and Wei Lin. 2021. EasyTransfer: A Simple and Scalable Deep Transfer Learning Platform for NLP Applications. In *CIKM*. ACM, 4075–4084.

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

[26] 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 (2020), 140:1–140:67.

[27] Timo Schick and Hinrich Schütze. 2021. Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference. In *EACL*. 255–269.

[28] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. In *NIPS*. 4077–4087.

[29] Jianlin Su, Ahmed Murtadha, Shengfeng Pan, Jing Hou, Jun Sun, Wanwei Huang, Bo Wen, and Yunfeng Liu. 2022. Global Pointer: Novel Efficient Span-based Approach for Named Entity Recognition. *CoRR* abs/2208.03054 (2022).

[30] Austin Cheng-Yun Tsai, Sheng-Ya Lin, and Li-Chen Fu. 2022. Contrast-Enhanced Semi-supervised Text Classification with Few Labels. In *AAAI*. 11394–11402.

[31] 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. In *EMNLP Workshop BlackboxNLP*.

[32] Chengyu Wang, Minghui Qiu, Taolin Zhang, Tingting Liu, Lei Li, Jianing Wang, Ming Wang, Jun Huang, and Wei Lin. 2022. EasyNLP: A Comprehensive and Easy-to-use Toolkit for Natural Language Processing. *CoRR* abs/2205.00258 (2022).

[33] Jianing Wang, Wenkang Huang, Minghui Qiu, Qiuhui Shi, Hongbin Wang, Xiang Li, and Ming Gao. 2022. Knowledge Prompting in Pre-trained Language Model for Natural Language Understanding. In *EMNLP*. Association for Computational Linguistics, 3164–3177.

[34] Jianing Wang, Chengyu Wang, Jun Huang, Ming Gao, and Aoying Zhou. 2023. Uncertainty-aware Self-training for Low-resource Neural Sequence Labeling. *CoRR* abs/2302.08659 (2023).

[35] Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021. Entailment as Few-Shot Learner. *CoRR* abs/2104.14690 (2021).

[36] Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D.Q. Bui, Junnan Li, and Steven C. H. Hoi. 2023. CodeT5+: Open Code Large Language Models for Code Understanding and Generation. *arXiv preprint* (2023).

[37] Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. LiST: Lite Prompted Self-training Makes Parameter-efficient Few-shot Learners. In *NAACL*. 2262–2281.

[38] Yue Wang, Weishi Wang, Shafiq R. Joty, and Steven C. H. Hoi. 2021. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation. In *EMNLP*. Association for Computational Linguistics, 8696–8708.

[39] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models are Zero-Shot Learners. In *ICLR*. OpenReview.net.

[40] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=_VjQlMeSB_J

[41] Zhenyu Wu, Yaoxiang Wang, Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Jingjing Xu, and Yu Qiao. 2023. OpenICL: An Open-Source Framework for In-context Learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Association for Computational Linguistics, Toronto, Canada, 489–498.

[42] Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaoweihua Liu, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. CLUE: A Chinese Language Understanding Evaluation Benchmark. In *COLING*, Donia Scott, Núria Bel, and Chengqing Zong (Eds.). 4762–4772.

[43] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *NeurIPS*. 5754–5764.

[44] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *ACL*. Association for Computational Linguistics, 1–9.

[45] Taolin Zhang, Chengyu Wang, Nan Hu, Minghui Qiu, Chengguang Tang, Xiaofeng He, and Jun Huang. 2022. DKPLM: Decomposable Knowledge-Enhanced Pre-trained Language Model for Natural Language Understanding. In *AAAI*. AAAI Press, 11703–11711.

[46] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *ICML (Proceedings of Machine Learning Research, Vol. 139)*. PMLR, 12697–12706.