OpenICL: An Open-Source Framework for In-context Learning

Zhenyu Wu♦†∗, Yaoxiang Wang♣†∗, Jiacheng Ye♠†
Jiangtao Feng◇, Jingjing Xu◇, Yu Qiao◇, Zhiyong Wu◇‡
◇Shanghai AI Laboratory   ♦ East China Normal University
♣Xiamen University   ♠The University of Hong Kong
carsonye@cs.hku.hk, {wuzhenyu,wangyaoxiang}@pjlab.org.cn
{fengjiangtao,xujingjing,qiaoyu,wuzhiyong}@pjlab.org.cn

Abstract

In recent years, In-context Learning (ICL) has gained increasing attention and emerged as the new paradigm for large language model (LLM) evaluation. Unlike traditional fine-tuning methods, ICL instead adapts the pre-trained models to unseen tasks *without* any parameter updates. However, the implementation of ICL is sophisticated due to the diverse retrieval and inference methods involved, as well as the varying pre-processing requirements for different models, datasets, and tasks. A unified and flexible framework for ICL is urgently needed to ease the implementation of the aforementioned components. To facilitate ICL research, we introduce OpenICL, an open-source toolkit for ICL and LLM evaluation. OpenICL is research-friendly with a highly flexible architecture that users can easily combine different components to suit their needs. It also provides various state-of-the-art retrieval and inference methods to streamline the process of adapting ICL to cutting-edge research. The effectiveness of OpenICL has been validated on a wide range of NLP tasks, including classification, QA, machine translation, and semantic parsing. As a side-product, we found OpenICL to be an efficient yet robust tool for LLMs evaluation. OpenICL is released at `https://github.com/Shark-NLP/OpenICL`.

1 Introduction

The rise of large language models (LLMs) (Brown et al., 2020; Zhang et al., 2022a; Scao et al., 2022) has shown impressive emergent In-Context Learning (ICL) ability (Wei et al., 2022a). Different from finetuning which requires parameter updates, ICL can perform inference with model parameters frozen. ICL sidesteps the resource-intensive nature of fine-tuning, yet still yields comparable results

to fine-tuned models in specific tasks (Zhao et al., 2021; Lu et al., 2022; Gao et al., 2021a). However, we observed a lack of a unified framework for ICL. Implementations from existing projects are often high-customized to their own needs, thus making further development and comparisons with previous approaches a challenge.

The basic ICL pipeline contains two steps: retrieval and inference. Given a testing input X', in the retrieval stage, several examples from the training set are retrieved as in-context demonstrations. In the inference stage, these demonstrations are prepended before X' and fed into the LLM to generate the prediction. Researchers have explored various methods for both retrieval(e.g., BM25 (Robertson and Zaragoza, 2009), TopK (Liu et al., 2022; Gao et al., 2021a) and VoteK (Su et al., 2022)) and inference(e.g., perplexity-based (Brown et al., 2020), channel-based (Min et al., 2022), and Chain-of-thoughts (Wei et al., 2022b)). However, these methods are often implemented under different frameworks, and/or evaluated using different LLMs and tasks. These inconsistencies make systematic evaluations and comparisons of various methods challenging, thus hindering the development of better ICL methods.

To address this issue, we present OpenICL, an open-source and easy-to-use toolkit for ICL. OpenICL has many state-of-the-art retrieval and inference methods built in to facilitate systematic comparison and fast research prototyping. OpenICL also provides a unified and flexible interface for the development and evaluation of new ICL methods. Users can easily incorporate different retrieval and inference methods, as well as different prompt instructions, into their pipelines. To validate OpenICL's implementation and design, we use OpenICL to evaluate LLMs on several NLP tasks, including classification, question answering, translation, and semantic parsing. Our contributions are summarized as follows:

∗Work done while interning at Shanghai AI Lab.
†Equal Contribution.
‡Corresponding Author.

- We propose OpenICL, an easy-to-use and extensible ICL framework for zero-/few-shot evaluation of language models

- OpenICL provides a wide range of ICL methods, LLMs, and tasks, requiring as little as a few lines of code to use and paving the way for more extensions in the future.

- We provide complete tutorials to walk users through the framework, thus facilitating research and development of ICL.

2 Related Work

In-context Learning   Besides the classic "pre-train and fine-tune" paradigm, Brown et al. (2020) proposed In-context learning (ICL), a new paradigm that leverages pre-trained language models to perform new tasks without any gradient-based training. It appends a small number of training examples as prompts before the test input, and have shown to be able to improve LLMs' performance in few-shot scenarios and generalize to a wide range of downstream tasks, such as information retrieval (Tay et al., 2022), fact checking (Rae et al., 2021), commonsense reasoning (Geva et al., 2021), arithmetic reasoning (Cobbe et al., 2021), machine trainslation (Agrawal et al., 2022; Lin et al., 2021a), and data generation (Ye et al., 2022), etc.

Aside from those early successes, researchers have developed more sophisticated ICL methods that require some intermediate reasoning steps. Among them, chain-of-thoughts (CoT) is the first attempt that significantly surpasses the previous state-of-the-art methods on many reasoning tasks (Wei et al., 2022b). After that, different variants of CoT have been proposed to strengthen its performance, such as Self-Ask (Press et al., 2022), iCAP (Wang et al., 2022), Least-to-Most prompting (Zhou et al., 2022), and Selection-Inference (Zhang et al., 2022b; Fu et al., 2022).

Despite the surprising performance, ICL has been criticized for being very sensitive to the choice and ordering of in-context examples (Zhao et al., 2021; Lu et al., 2022). To address this problem, different criterion and context construction methods have been proposed. Gao et al. (2021a) and Liu et al. (2022) select examples that are closer to the test input in the embedding space; a line of work (Su et al., 2022; Levy et al., 2022; Ye et al., 2023) select the most representative examples in the training set to encourage diversity of in-context examples; Wu et al. (2022) observe that Minimum Description Length (MDL) principle can be an effective criterion for in-context example selection.

Prompt Learning   Prompt learning (Liu et al., 2021) is a special case of ICL without any in-context examples. Prompt learning comprises various topics including manual template engineering (Petroni et al., 2019; Brown et al., 2020), automated template learning (Wallace et al., 2019; Shin et al., 2020; Li and Liang, 2021), and answer engineering (Gao et al., 2021b; Schick and Schütze, 2021). We refer the readers to the usage of OpenPrompt (Ding et al., 2021) which is a toolkit specially designed for prompt learning. In comparison, OpenICL focuses more on integrating various exemplar retrieving approaches and inference strategies for in-context learning. Note that OpenICL can also seamlessly support prompt learning by setting the number of in-context examples to zero and specifying the manual or pre-searched prompt templates by OpenPrompt for different tasks.

3 OpenICL

In this section, we first explain OpenICL's design principles. Then, we will briefly describe OpenICL's two major components, namely, the `Retriever` and `Inferencer`.

3.1 Design Principles

The design principle of OpenICL is to facilitate in-context learning research and enable efficient and robust large language model evaluation. In detail, we consider the following principles:

[P1: Modularity]   Since ICL is a fast-evolving research field, the design of OpenICL should be decoupled such that different components can be easily modified to support latest methods and/or combined to suit various tasks and application needs.

[P2: Efficiency]   Nowadays, large language models can have hundreds of billions of parameters. To support inference at such a massive scale, OpenICL should be optimized to enable efficient parallel inference.

[P3: Generality]   ICL has been widely used in all fields in NLP, so OpenICL needs a flexible interface that enables it to work with various LLMs, tasks, retrieval methods, and inference approaches.
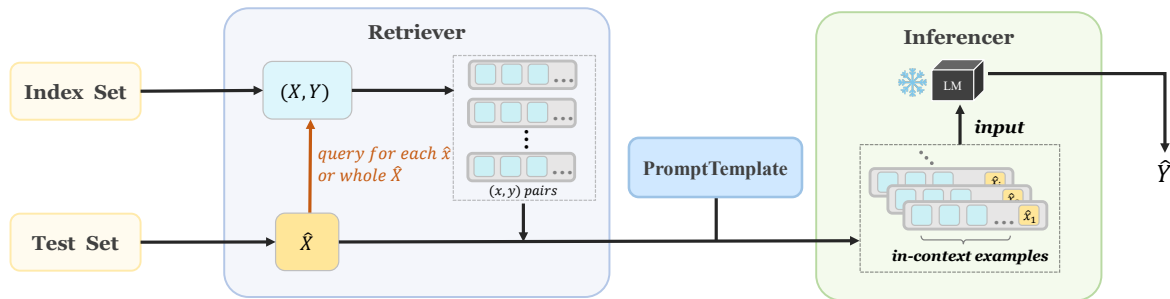
Figure 1: Overview of the architecture in OpenICL. OpenICL first obtains proper in-context examples from an index set for each test input or for the whole test set via retrieval methods (e.g., TopK or VoteK) specified by the users. Then the in-context examples and test input are concatenated into a single sequence based on the provided prompt template. Finally, all the prompts are fed into the language model to infer the output through defined inference strategies (e.g., Chain-of-thought).

3.2 Architecture Overview

Figure 1 overviews OpenICL's architecture. For each input $\hat{x}$ from the test set $\hat{X}$, the Retriever retrieves several (x, y) pairs (represented as one row in the dashed box) from an index set (X, Y) as $\hat{x}$'s in-context examples. These examples, as well as $\hat{x}$, are then formatted according to the user-defined prompt template and concatenated to form a text sequence. After that, the Inferencer digests these sequences and fed them into the LLMs to obtain the model prediction $\hat{Y}$.

3.3 Modularity

To satisfy Principle P1, OpenICL adopts a loosely-coupled design between components. These components separate the data pre-processing, retrieval, and inference processes with very flexible interfaces that allow easy customization to fit specific needs. Two major components are detailed below:

Retriever Retriever is responsible for retrieving in-context examples from the pre-existing training data. This module supports both corpus-level (i.e., only retrieving one group of examples for the whole test set) and instance-level (i.e., retrieving examples for each testing input individually) retrieval methods. OpenICL primarily supports learning-free retrieval methods as follows:

- Random: Early practice (Brown et al., 2020) of ICL often randomly select examples to construct the context. Although Random brings high variance for ICL performance, it is still the popular choice when there are only a few demonstrations available (Wei et al., 2022b; Zhao et al., 2021).

- Heuristic method: To overcome the disadvantage of Random, various semantic similarity based retrieval methods have been proposed and shown great promise, such as BM25 (Robertson and Zaragoza, 2009), TopK (Liu et al., 2022; Gao et al., 2021a), and VoteK (Su et al., 2022).

- Model-based method: More recently, researchers have explored using models' confidence in the output to select and order examples, such as entropy (Lu et al., 2022) and MDL (Wu et al., 2022).

OpenICL has implemented the existing methods above to facilitate future research and systematic comparison. Furthermore, the flexibility of the Retriever module allows practitioners to select the retrieval method and make further modification that best suits their task and data. The interface of Retriever also allows users to pack those in-context examples and use them somewhere else.

Inferencer Inferencer invokes the pre-trained language model to generate predictions based on the concatenation of in-context examples and testing input. The Inferencer supports various inference methods:

- Direct: Brown et al. (2020) use tokens in the vocabulary to represent candidate answers and select the final prediction using the one with the highest probability.

- Perplexity: (Brown et al., 2020) compute the sentence perplexity of the sequence concatenation of input and candidate answers and

select the final prediction using the one with the lowest perplexity.

- Channel: Min et al. (2022) proposed to utilize channel models (Yu et al., 2016; Yee et al., 2019) to compute the conditional probability in a reversed direction, i.e., estimating the likelihood of input query given the label.

The flexibility of `Inferencer` also allows users to recursively invoke it to support multi-stage ICL methods, such as chain-of-thought (Wei et al., 2022b) and selection-inference (Creswell et al., 2022). Additionally, `Inferencer` can be augmented with a scorer to evaluate its prediction.

3.4 Efficiency

To satisfy Principle P2, we equip OpenICL with various parallelism techniques to enable efficient inference for large-scale models.

Data Parallel   Data parallel (Li et al., 2020) is a common technique used in parallel computing to improve the efficiency of large-scale computation tasks. OpenICL implements data parallelism to improve the performance of both the retrieval and inference steps. During retrieval and inference, data is divided into smaller batches for processing. Additionally, for models that can fit into GPU's VRAM, OpenICL implements data parallelism by sharding the data across multiple GPUs and performing parallel inference on each GPU with a complete copy of the model. This significantly increases the inference speed when working with large datasets.

Model Parallel   In the era of LLMs, models often have billions or hundreds of billions of parameters that exceed modern GPUs' capacity. To handle this problem, we resort to model parallel (Shoeybi et al., 2019): a parallel computing technique that divides a large deep learning model into smaller sub-models, each of which can be run on a separate GPU. OpenICL supports model parallelism that users can easily parallelize their models with minimal modification to the code. Currently, we support Megatron (Shoeybi et al., 2019) and Zero (Rajbhandari et al., 2019).

3.5 Generality

To satisfy Principle P3, OpenICL is designed to maximize users' productivity by supporting a wide range of models, tasks, and methods:

[Model]   OpenICL supports both decoder-only LMs (e.g., GPT family (Radford and Narasimhan, 2018; Radford et al., 2019; Black et al., 2021; Wang and Komatsuzaki, 2021; Black et al., 2022), and encoder-decoder-based LMs (e.g., T5 (Raffel et al., 2020)). We also provide two alternatives for accessing the model: users can directly load model checkpoints for evaluation or access a model via API (e.g., OpenAI's GPT-3 series models; Brown et al. 2020; Chen et al. 2021; Ouyang et al.).[1]

[Tasks]   With the help of OpenICL, users can easily conduct experiments on both classification and generation tasks. OpenICL integrates HuggingFace's `datasets`[2] such that users can access and download thousands of NLP tasks with ease.

[Methods]   As aforementioned, OpenICL provides broad support for ICL methods that cover both retrieval and inference.   Furthermore, OpenICL offers the flexibility to return the results of the `Retriever` and `Inferencer` in a step-by-step manner, making it easy to integrate these intermediate results into other projects.

4   Toolkit Walkthrough

In this section, we demonstrate OpenICL by walking readers through several typical ICL use cases.

Example 1.   We first demonstrate how to use OpenICL to develop a typical ICL pipeline for language classification using a few lines of code and conduct evaluation on the popular sentiment classification dataset SST-2 (Socher et al., 2013). As shown in Figure 2, the pipeline begins with a `DatasetReader` which loads the dataset given its name on HuggingFace Dataset Hub[3] or local file path.   Users need to specify the names of columns where the input ("*text*") and output ("*label*") are stored.   Secondly, a customized `PromptTemplate` is instantiated with a dictionary that defines the prompts for each class label. The placeholder </E> and </Q> will be replaced by in-context examples and testing input, separately.   After that, we initiate the retriever based on TopK (Liu et al., 2022) and set the number of in-context examples to 8 ("*ice_num* = 8"). We select perplexity-based method to initiate the inferencer and use GPT2-XL as the LLM. Having

---

[1]https://openai.com/api/
[2]https://github.com/huggingface/datasets
[3]https://huggingface.co/datasets

```
1  from openicl import DatasetReader, PromptTemplate
2  from openicl import TopkRetriever, PPLInferencer, AccEvaluator
3
4  # Load dataset
5  data = DatasetReader('gpt3mix/sst2', input_columns=['text'], output_column='label')
6
7  # Define the prompt template for the task
8  tp_dict = { 0: '</E> Positive Movie Review: </Q>',
9              1: '</E> Negative Movie Review: </Q>' }
10 template = PromptTemplate(tp_dict, {'text':'</Q>'}, ice_token='</E>')
11
12 # Initiate the retriever and inferencer
13 retriever = TopkRetriever(data, ice_num=8)
14 inferencer = PPLInferencer(model_name='gpt2-xl')
15
16 # Run inference and calculate score
17 predictions = inferencer.inference(retriever, ice_template=template)
18 score = AccEvaluator().score(predictions=predictions, references=data.references)
```

Figure 2: Illustration of Example 1 which evaluates the ICL performance of GPT2-XL (1.5B) on SST-2 dataset with PPL inference strategy.

```
1  from datasets import load_dataset
2  from openicl import DatasetReader, PromptTemplate
3  from openicl import RandomRetriever, GenInferencer, BleuEvaluator
4
5  dataset = load_dataset("wmt16", 'de-en').map(lambda example: example['translation'])
6
7  data = DatasetReader(dataset, input_columns=['de'], output_column='en')
8
9  template = PromptTemplate('</E> German:</German> \n English: </English>',
10                           {'de':'</German>', 'en':'</English>'}, ice_token='</E>')
11
12 retriever = RandomRetriever(data, ice_num=8)
13
14 # Inference by direct generation
15 inferencer = GenInferencer(model_name='facebook/xglm-7.5B')
16 predictions = inferencer.inference(retriever, ice_template=template)
17
18 # calculate Bleu
19 score = BleuEvaluator().score(predictions=predictions, references=data.references)
```

Figure 3: Illustration of Example 2 that evaluates the ICL performance of XGLM (7.5B) on WMT16 (de-en) dataset with direct inference strategy.

```
1  from openicl import DatasetReader, PromptTemplate, BM25Retriever, CoTInferencer
2
3  data = DatasetReader('gsm8k', name='main',
4                       input_columns=['question'], output_column='answer')
5
6  template = PromptTemplate('</E> Question: </Q> \n Answer: </A>',
7                           {'question':'</Q>', 'answer':'</A>'},
8                           ice_token='</E>')
9
10 retriever = BM25Retriever(data, ice_num=4)
11
12 # Inference by Chain-of-Thought
13 cot_list=["Let's think step by step.",
14          "\nTherefore, the answer (arabic numerals) is"]
15
16 inferencer = CoTInferencer(cot_list=cot_list, api_name='gpt3')
17 predictions = inferencer.inference(retriever, ice_template=template)
```

Figure 4: Illustration of Example 3, which evaluates the ICL performance of *text-davinci-003* version of GPT-3 (175B) on GSM8K dataset with Chain-of-thought inference strategy.
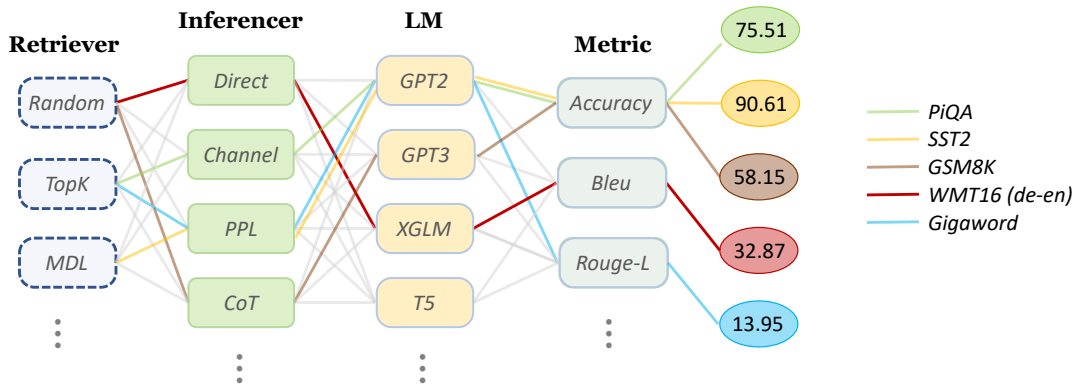
Figure 5: Evaluation results. We conduct experiments on five representative tasks with OpenICL and use different retrievers, inferencers, language models, and other components. In terms of model usage, we adopt GPT-Neo (2.7B) for SST2, PiQA, and Gigaword, XGLM (7.5B) for WMT16 (de-en), and *text-davinci-003* version of GPT-3 (175B) for GSM8K.

all these been set, we can run the inference by invoking the inferencer (line 17) and calculating the accuracy of the model's prediction(line 18).

Example 2. Figure 3 shows how to use OpenICL to work with generation problems. We consider the popular machine translation dataset WMT16 (Bojar et al., 2016). As in Example 1, we can easily load the dataset, define the prompt template, and initiate the retriever, by feeding new parameters to the function, respectively. The major API difference from Example 1 is that (i) we add some pre-processing for the translation task (line 5); (ii) *PPLInferencer* is replaced by inferencer tailored for generation (line 16); (iii) we use BLEU to evaluate model performance.

Example 3. OpenICL also supports more advanced ICL methods, as shown in Figure 4. Users can seamlessly switch to CoT by only modifying two lines of code: line 14 defines the template for CoT and line 15 initiates the inferencer with GPT3 using OpenAI's API. Similar multi-step ICL methods such as Self-Consistency (Wang et al., 2022) and Selection-Inference (Creswell et al., 2022) can also be easily implemented by inheriting the superclass `Inferencer` designed in OpenICL.

5 Evaluation

To demonstrate OpenICL's flexibility we conducted experiments on a diverse set of datasets, LLMs, and ICL methods. We consider PiQA (Bisk et al., 2019) for commonsense reasoning, SST-2 (Socher et al., 2013) for sentiment analysis, GSM8K (Cobbe et al.,

2021) for arithmetic reasoning, WMT16 de-en (Bojar et al., 2016) for machine translation and Gigaword (Napoles et al., 2012) for summarization. We've also tested various LLMs, including GPT-Neo (2.7B) (Black et al., 2021; Gao et al., 2020), *text-davinci-003* version of GPT-3 (175B), and XGLM (7.5B) (Lin et al., 2021b). We use OpenAI's official API[4] to access GPT-3. The detailed setups and results are shown in Figure 5. As we can see, components of OpenICL can be easily chained to support different evaluation needs and replicate results of state-of-the-art methods.

6 Conclusion

We present OpenICL, an open-source toolkit for In-context learning. OpenICL provides a convenient and flexible interface for in-context learning practice and research. Our modular design allows it to support a wide range of LLMs, tasks, and ICL methods with ease. We implement both model parallelism and data parallelism to make inference of large models more efficient. OpenICL is highly extensible, and we will continue to update it to keep pace with the latest research. Despite the promising results, ICL is still in its early stages, and many challenges remain. We believe OpenICL will be a valuable resource for researchers and practitioners to facilitate their research and development.

---

[4]https://openai.com/api/

References

Sweta Agrawal, Chunting Zhou, Mike Lewis, Luke Zettlemoyer, and Marjan Ghazvininejad. 2022. In-context examples selection for machine translation.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. PIQA: Reasoning about Physical Commonsense in Natural Language. *arXiv e-prints*, page arXiv:1911.11641.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, USVSN Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. Gpt-neox-20b: An open-source autoregressive language model.

Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.

Ondřej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Liane Guillou, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Aurélie Névéol, Mariana Neves, Pavel Pecina, Martin Popel, Philipp Koehn, Christof Monz, Matteo Negri, Matt Post, Lucia Specia, Karin Verspoor, Jörg Tiedemann, and Marco Turchi, editors. 2016. *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*. Association for Computational Linguistics, Berlin, Germany.

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. *CoRR*, abs/2005.14165.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-Inference: Exploiting Large Language Models for Interpretable Logical Reasoning. *arXiv e-prints*, page arXiv:2205.09712.

Ning Ding, Shengding Hu, Weilin Zhao, Yulin Chen, Zhiyuan Liu, Hai-Tao Zheng, and Maosong Sun. 2021. OpenPrompt: An Open-source Framework for Prompt-learning. *arXiv e-prints*, page arXiv:2111.01998.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. *CoRR*, abs/2210.00720.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021a. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021b. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did Aristotle Use a Laptop? A Question Answering Benchmark with Implicit Reasoning Strategies. *arXiv e-prints*, page arXiv:2101.02235.

Itay Levy, Ben Bogin, and Jonathan Berant. 2022. Diverse demonstrations improve in-context compositional generalization. *arXiv preprint arXiv:2212.06800*.

Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, and Soumith Chintala. 2020. Pytorch distributed: Experiences on accelerating data parallel training.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona T. Diab, Veselin

Stoyanov, and Xian Li. 2021a. Few-shot learning with multilingual language models. *CoRR*, abs/2112.10668.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, Ramakanth Pasunuru, Sam Shleifer, Punit Singh Koura, Vishrav Chaudhary, Brian O'Horo, Jeff Wang, Luke Zettlemoyer, Zornitsa Kozareva, Mona T. Diab, Veselin Stoyanov, and Xian Li. 2021b. Few-shot learning with multilingual language models. *CoRR*, abs/2112.10668.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for gpt-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Noisy channel language model prompting for few-shot text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5316–5330, Dublin, Ireland. Association for Computational Linguistics.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX)*, pages 95–100, Montréal, Canada. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*,

pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *CoRR*, abs/2210.03350.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling Language Models: Methods, Analysis & Insights from Training Gopher. *arXiv e-prints*, page arXiv:2112.11446.

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.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2019. Zero: Memory optimization towards training A trillion parameter models. *CoRR*, abs/1910.02054.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman

Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv preprint*, abs/2211.05100.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *arXiv e-prints*, page arXiv:1909.08053.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. 2022. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975*.

Yi Tay, Vinh Q. Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer Memory as a Differentiable Search Index. *arXiv e-prints*, page arXiv:2202.06991.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Boshi Wang, Xiang Deng, and Huan Sun. 2022. Iteratively prompt pre-trained language models for chain of thought. In *The 2022 Conference on Empirical Methods for Natural Language Processing*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *arXiv e-prints*, page arXiv:2203.11171.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2022. Self-adaptive in-context learning.

Jiacheng Ye, Jiahui Gao, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2022. ProGen: Progressive zero-shot dataset generation via in-context feedback. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3671–3683, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional exemplars for in-context learning. *arXiv preprint arXiv:2302.05698*.

Kyra Yee, Nathan Ng, Yann N. Dauphin, and Michael Auli. 2019. Simple and effective noisy channel modeling for neural machine translation. *CoRR*, abs/1908.05731.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomás Kociský. 2016. The neural noisy channel. *CoRR*, abs/1611.02554.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022a. Opt: Open pre-trained transformer language models.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022b. Automatic chain of thought prompting in large language models. *CoRR*, abs/2210.03493.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-Shot Performance of Language Models. *arXiv e-prints*, page arXiv:2102.09690.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *ArXiv preprint*, abs/2205.10625.