UNIFIT: A Unified Framework For Instruction Tuning To Improve Instruction Following Ability For Large Language Models

Qiang Huang♠,Feng Huang♠,Dehao Tao◇ , Bingkun Wang♣,Yongfeng Huang‡†△[1]
♠ School of Computer Science and Technology,Xinjiang University ,China
‡ Department of Electronic Engineering, Tsinghua University,China
♣ School of Information Engineering, Zhengzhou College of Finance and Economics,China
◇ School of Humanities, Tsinghua University,China
† Zhongguancun Lab,China

Abstract

Extensive instruction tuning of large language models(LLMs) has proven to be a powerful technique, extending the outstanding performance of general LLMs to new tasks. Consequently, the integration of state-of-the-art(SOTA) general open-source models with specific domains leverages instruction tuning to unlock the emerging capabilities of large language models(LLMs) in those domains. Current practices in instruction tuning often rely on expanding the size of the dataset without a clear strategy to ensure data quality. This can inadvertently introduce noise and degrade model performance. Furthermore, there is currently no unified approach for the quantity, quality, and diversity of instruction tuning data, as well as the methods for instruction tuning. As a result, this severely hampers the practicality and universality of instruction tuning.
Addressing these issues, we propose a **UNI**fied **F**ramework for **I**nstruction **T**uning(**UNIFIT**), namely Concept Tree generation for instruction tuning data, instruction following difficulty selecting data for high-quality instruction tuning, and the incorporation of random noise embeddings(NE) to enhance model performance during tuning. Through experiments involving multiple models, domains, and orders of magnitude, our proposed instruction tuning framework not only enhances the diversity of instruction tuning data but also achieves a remarkable 60% reduction in training time consumption, with a mere 6% of all instruction tuning data, surpassing the performance of using all instruction tuning data by 11%. This universally applicable instruction tuning framework signifies a substantial advancement in the generality of large language model instruction tuning, marking a revolutionary leap forward and promising efficiency gains while being resource-conscious.

Keywords: Large Language Models, Instruction Tuning, High-Quality Instruction Data

Introduction

The large language models(LLMs) have fundamentally changed the landscape of artificial intelligence development. Prominent models such as ChatCPT, GPT-4, LLaMa2(Touvron, Martin, et al., 2023), Baichuan2(Yang, Xiao, et al., 2023), Qwen(Bai, Bai, et al., 2023), and Mistral(Jiang et al., 2023) showcase advanced text understanding and generation capabilities through the utilization of extensive datasets and sophisticated training methodologies. Applications for these models across different domains, including interactive systems, automated content generation, and support for scientific enquiry(Shanahan, McDonell, & Reynolds, 2023).

To optimize the performance of LLMs in specific tasks or domains and make their output adapt to particular contexts or instructions is imperative. The technique employed for achieving this goal is known as instruction tuning. Its operational approach involves providing explicit training instructions to LLMs to generate outputs that are more consistent with the expected results. Carefully crafted instructions or prompts offer crucial background information, enhancing the capability of model to generate task-specific outputs. This method strengthens the generalization of the model to new tasks and reinforces its alignment with user-defined objectives. Instruction tuning is an indispensable step that not only activates the valuable knowledge acquired by LLMs during training but also enables them to interact with humans in a natural conversational manner. Presently, we are faced with the dilemma that more and more tasks require the application of the general state-of-the-art(SOTA) LLMs through instruction tuning to specific domains, such as medicine, law, and food, which are closely related to human life. For workers in these domains, who have little or no background in computer technology, the task of instruction tuning is almost impossible, combining the instruction data generation involved in the process, the selection of high-quality instruction data, and effective instruction tuning methods. This dilemma severely hampers the application of LLMs in vertical domains. To solve this problem, there is an urgent requirement for a unified framework for effective instruction tuning to assist those domain practitioners to be able to complete the task of vertical domain application in a few simple steps. This will greatly improve the practicality and effectiveness of instruction tuning and reduce the difficulties of applying LLMs to vertical domains. It further promises that the SOTA general LLMs can be effectively applied in various domains to enhance work efficiency, promote scientific research, and improve the quality of life.

Traditionally, instruction tuning has predominantly relied on accumulating extensive datasets. In the pre-era of LLMs, the requisite data for instruction tuning primarily depended on manual composition.PROMPTSOURCE(Bach, Sanh, et al., 2022) and SUPERNATURALINSTR
-UCTIONS(Wang, Mishra, et al., 2022) are two noteworthy datasets that use extensive manual annotation to collect instructions to construct the T0 and TK-INSTRUCT, which

[1]△ denotes corresponding author

is however expensive due to the significant amount of manual labor consumed. In the era of LLMs, a notable endeavor is the Self-Instruct(Wang et al., 2023) framework, which leverages minimal human annotation to generate substantial data for instruct tuning. However, the Self-Instruct(Wang et al., 2023) framework often grapples with limitations in terms of limited data diversity, hindering its ability to cover a truly diverse array of vertical application tasks. This struggle becomes particularly evident in practical application scenarios. From LIMA(Zhou et al., 2023), a pioneering insight emphasizes the art of instruction tuning: it is the quality of the data that determines the performance of a model, not the mass of data. Findings of LIMA(Zhou et al., 2023) emphasize that even a limited amount of hand-managed, high-quality data can improve a model's instructional adherence ability. This site presents both an opportunity and a challenge. While it highlights the validity of high-quality data, the question of how to automatically identify high-quality data from large available datasets remains under-researched, especially when combined with effective instructional tuning methods during tuning.

To bridge the above gaps, we propose a three-stage instruction tuning framework(See Fig2). First, we use a Concept Tree(See Fig1) to automatically generate instruction tuning data. Specifically, we take the root node in the tree structure as the domain to which the instruction tuning will be applied, and the nodes in the second level of the tree as the scope involved in the current domain; the root node and the nodes in the second level of the tree are the ones that require human intervention to determine. Subsequent nodes will use the hints to generate sub-concepts iteratively until the concept hierarchy satisfies the domain and scope, forming a complete concept tree. The leaf nodes of the concept tree are used to generate specific instruction data in conjunction with the cueing macromodel. Second, to realize the quest for automatic identification of high-quality data, we designed to train a scoring model by randomly extracting K instructions from each leaf node in the concept tree, and the data used to train the scoring model typically requires only 5% of all data. Using the scoring model in combination with the instruction-following difficulty scores to automatically filter out some high-quality data from all the data, we call it Pithy data, which is typically 5%-10%. Finally, we added random noise to the embedding vectors during instruction training and obtained significant improvements by including this simple application during tuning.

We demonstrate on extensive experiments across multiple models, domains, and orders of magnitude that our proposed instruction tuning framework not only expands the diversity of instruction tuning data but also achieves a win rate of only 6% with all instruction tuning data over 11% with all instruction tuning data at a savings of 60% in training time consumption.UNIFIT marks the versatility of instruction tuning for LLMs that has been greatly improved, gaining efficiency and resource-awareness advances and ushering in a transformative leap in instruction tuning.

Through extensive experiments across various models, domains, and scales, we have validated the effectiveness of our proposed instruction tuning framework. UNIFIT not only enhances the diversity of instruction tuning data but also, with a mere 6% utilization of all instruction tuning data, achieves a win rate exceeding that of using the entire dataset by 11%, while concurrently reducing training time consumption by 60%. UNIFIT marks a significant advancement in the universality of instruction tuning for LLMs, showcasing progress in efficiency and resource utilization, and marking a transformative leap in the field of instruction tuning.

The main contributions of this paper:

1. We employ a self-guided concept tree to autonomously generate instructional data, resulting in a more diverse dataset compared to the baseline Self-Instruct approach.

2. We introduce the UNIFIT, establishing a comprehensive pipeline for the entire instruction fine-tuning process. This significantly enhances the practicality and generality of instruction tuning, encompassing three main components: data generation for instruction tuning, selection of instructional tuning data, and effective methods for instruction tuning.

3. UNIFIT has been empirically validated through extensive instruction tuning experiments, demonstrating its effectiveness. Even when utilizing only 6% of all instruction tuning data while conserving 60% of resources, it achieved a win rate surpassing 11% compared to using the entire dataset.

Methodology

In pursuit of achieving efficient reuse and robust generalization capabilities for instruction tuning, UNIFIT aims to address three key challenges: ensuring the diversity of instruction data, effectively automating the selection of high-quality instruction data, and enhancing the effectiveness of instruction tuning.

Instruction Tuning Data Generation

In our approach (See Fig1), we designate the first-layer nodes in the tree structure, specifically the root node, as the domain to which the instruction fine-tuning model will be applied. The value of the root node, representing the domain to which the instruction fine-tuning will be applied, is determined by downstream tasks. Examples of such domains include cybersecurity and public safety.

Subsequently, the second-layer nodes of the tree are used to define the scope of the current domain. Taking the cybersecurity domain as an example, the root node is set as "cybersecurity," and the second layer encompasses the five major infrastructures within the cybersecurity domain—namely, Email, Routing, CDN, DNS, and PKI. Therefore, the determination of the root node and the second-layer nodes requires manual input based on the specific application domain. Subsequent nodes are generated iteratively by prompting the LLM until the concept hierarchy

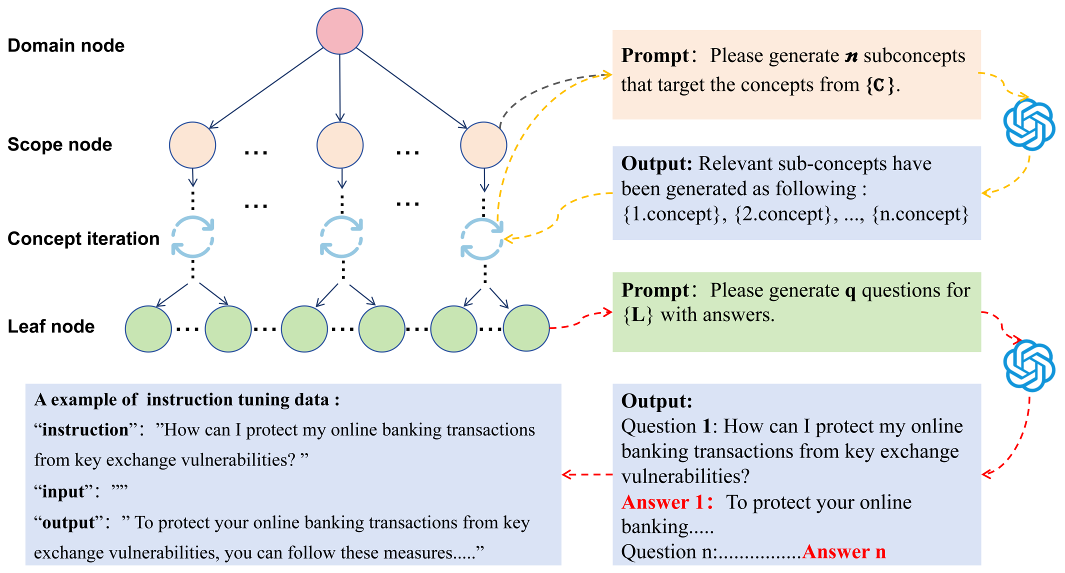
Domain node
Scope node
Concept iteration
Leaf node
Prompt: Please generate n subconcepts that target the concepts from {C}.
Output: Relevant sub-concepts have been generated as following : {1.concept}, {2.concept}, ..., {n.concept}
Prompt: Please generate q questions for {L} with answers.
Output:
Question 1: How can I protect my online banking transactions from key exchange vulnerabilities?
Answer 1: To protect your online banking.....
Question n:................Answer n
A example of instruction tuning data :
"instruction": "How can I protect my online banking transactions from key exchange vulnerabilities? "
"input": ""
"output": " To protect your online banking transactions from key exchange vulnerabilities, you can follow these measures....."

Figure 1: Framework of Conceptual Tree.Domain nodes represent the application domains for generating instruction data. Scope nodes indicate the scopes involved in this domain. Concept iteration involves generating child concepts from a scope node as the parent node. The newly generated child concept is then used as the parent concept to generate new child concepts, iterating continuously until the application domain requirements are met. Here, n represents the number of child concepts to generate,C denotes the current parent concept. q is the number of instances generated for the current leaf node, and L represents the current leaf node. This will always stay closely aligned to the application domain, maintaining higher relevance with a deeper depth of instruction data.

satisfies the requirements of the chosen domain and scope, resulting in the formation of a comprehensive concept tree. A prompt for generating sub-concepts is *Please generate n sub-concepts for C,* where n represents the number of sub-concepts, and C is the parent concept. Starting from the second layer, the generation of sub-concepts begins using the second-layer nodes as parent concepts. Subsequently, the newly generated sub-concepts are used as parent concepts in an iterative process to generate sub-concepts.

Following this, the leaf nodes of the concept tree, in conjunction with prompts, are utilized by the LLM to generate specific instances of instruction data. A prompt for generating data instances is *Please generate q questions for L and provide answers,* where q is the number of instances to be generated, and L represents the leaf node concept. Under the setting of the concept tree, it is even possible to utilize prompts to generate multiple multi-turn question-answer instances for each leaf node. As widely acknowledged, multi-turn data plays an extremely important role in enhancing the effectiveness of instruction fine-tuning. A prompt for generating multi-turn question-answer data is *Based on the question Q, generate a new question that is highly likely to be asked next, and provide the answer.*.

High Quality Instruction Data Selection

The process of filtering high-quality instruction data mainly consists of two steps(See Fig2). In the first step, we initially randomly extract K instances of instructions from each leaf node concept, forming the training data T for training the scoring model. Our numerous experiments indicate that T typically constitutes 5%-10% of all instruction instances, which is sufficient to train a scoring model capable of effectively completing the task. The value of K is determined based on the size of T. The goal of this stage is to force the model to first experience a subset of the target dataset, enabling the initial model to acquire basic instruction following ability.

In the second step inspired by From Quantity to Quality(Li et al., 2023), the scoring model trained in the first step is utilized to automatically filter high-quality instruction data based on the score of instruction following difficulty(IFD). The notion of instruction following difficulty refers to the disparity between the model's anticipated response and its autonomous generation ability. This metric evaluates the ability of model to generate an appropriate response based on the provided instruction. It measures the alignment of the model's output with the instruction and the corresponding correct answer. In the entire process of selecting Pithy data, our approach exhibits a fundamental distinction from the method proposed by From Quantity to Quality(Li et al., 2023). The instruction instance sets randomly sampled from each leaf node concept in our method can comprehensively cover the entire spectrum of instruction data categories, ensuring the effectiveness of the scoring model with minimal time consumption. In contrast, their approach necessitates

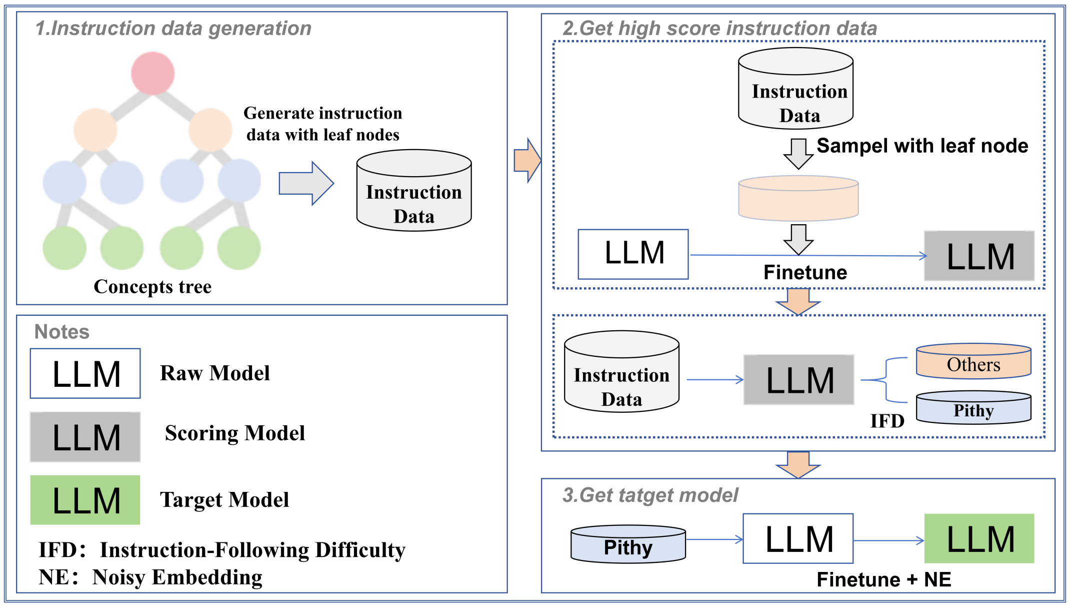

Figure 2: Framework of UNIFIT. It consists of three parts: instruction data generation using Concept Trees, filtering high-quality instruction data(Pithy) through instruction following difficulty(IFD), and instruction tuning with the addition of noise embedding(NE). Specific details are available at Methodology

the use of a language model ensemble and the K-Means algorithm for clustering to determine clusters. This not only consumes substantial computational resources and time but also fails to guarantee the impact of the cluster count on the scoring model, as the cluster count is a manually determined hyperparameter.

Noise Embedding Boosting Instruction Tuning

In pursuit of better instruction tuning results(See Fig2), we are inspired by NEFTune(Jain et al., 2023) to add random noise to the embedding vector of the training data during the forward fine-tuning process. Each step of NE begins by sampling an instruction from the dataset instructions and converting its labeling to an embedding vector. NE then departs from standard training by adding a random noise vector to the embedding. The noise is generated by sampling iid uniform entries, each in the range [-1,1], and then scaling the entire noise vector to $\alpha\sqrt{Ld}$, where L is the sequence length, d is the embedding dimension, and α is a tunable parameter. Significant results were obtained by this simple improvement of noise embedding(NE).

Experiment

Model

UNIFIT selected the four most popular models for the experiment. Due to computational resource constraints, the size of the models was chosen to be 7 billion parameters.

LLaMa2-7B(Touvron et al., 2023): The new open source LLM released by Meta, LLaMa2 is a continuation and upgrade of the Llama model.LLaMa2 has a 40% increase in the pre-training corpus compared to LLaMa, increasing to 2 trillion tokens, and the text sources in the training data are more diverse.

Mistral-7B(Jiang et al., 2023): Mistral-7B is an open-source LLM that has demonstrated amazing capabilities on a wide range of tasks.

Baichuan2-7B(Yang et al., 2023): Baichuan2 is a new generation of open source LLM launched by Baichuan Intelligence, which is trained with a high-quality corpus of 2.6 trillion Tokens.Baichuan2 has already achieved eye-catching results in various dimensions.

Qwen-7B(Bai et al., 2023): Qwen-7B was pre-trained on over 2.2 trillion tokens. On a wide variety of benchmarks tested, Qwen-7B typically outperforms existing open-source models and is comparable to performance on some LLMs.

Evaluation Metric

Evaluating the instruction-following capabilities of LLM poses a challenge. Despite extensive research dedicated to creating automated evaluation metrics for LLMs(Chang, Wang, et al., 2023), human judgment remains unparalleled. However, it is both labor-intensive and susceptible to human biases. Recent advancements in independent LLM evaluation(Cassano et al., 2023; Zheng, Chiang, et al., 2023) suggest that utilizing state-of-the-art(SOTA) LLMs for evaluation is an urgent and highly promising endeavor. In this paper, we undertake a comparative evaluation utilizing GPT-4.

Table 1: Time spent in tuning each model on the 500K dataset for the cybersecurity domain. Selection time consists of training the scoring model and filtering the Pithy data with the scoring model. Traintime is the time spent training the target model. The bolded text is what we get with our method. All numbers are in minutes.

	LLaMa2-7B				Mistral-7B				Baichan2-7B				Qwen-7B			
	Pithy	All	Pithy NE	All NE	Pithy	All	Pithy NE	All NE	Pithy	All	Pithy NE	All NE	Pithy	All	Pithy NE	All NE
Select time	2528	0	2528	0	2519	0	2519	0	2608	0	2608	0	2558	0	2558	0
Train time	193	8282	198	8298	188	8259	189	8263	205	8688	208	8693	195	8522	195	8525
All time	**2721**	8282	**2726**	8298	**2707**	8259	**2708**	8263	**2813**	8688	**2816**	8693	**2753**	8522	**2753**	8525

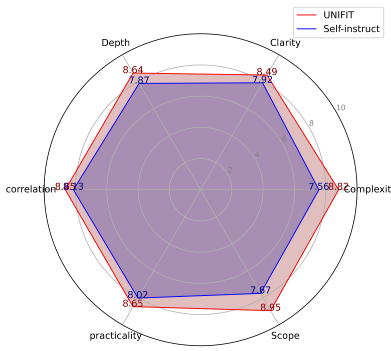

Figure 3: GPT-4 conducted a diversity evaluation on the instruction data from UNIFIT and Self-Instruct across six dimensions. The results of this evaluation are based on the 500K data in the domain of cybersecurity

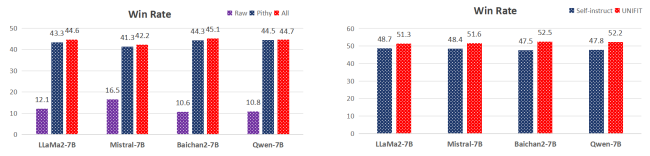

Figure 4: Results of the win rate in the evaluation of instruction-following ability were compared between the model tuned for All instruction data and the model tuned for Pithy data with the addition of noise embedding using GPT4 on a 500K dataset in the cybersecurity domain. It is worth noting that the Pithy instruction data used only 6instruction data.

Evaluation of the Diversity of Instruction Tuning Data

We employed ChatGPT and GPT-4 to score instruction data generated through both the concept tree method and the self-instruct method. The scoring encompassed six dimensions: scope, complexity, practicality, depth, clarity, and correlation. To obtain more accurate results and minimize differences caused by randomness, we conducted multiple random extractions on the instruction data generated by both methods. Subsequently, the obtained scores were averaged for each dimension.

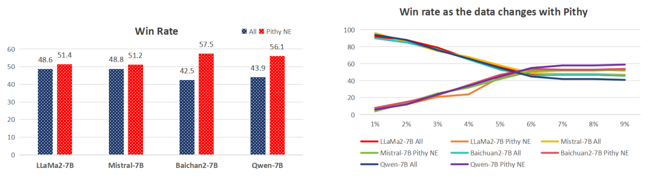

Figure 5: Results of the win rate in the evaluation of instruction-following ability was compared between the model tuned for All instruction data and the model tuned for Pithy data with the addition of noise embedding using GPT4 on a 500K dataset in the cybersecurity domain. It is worth noting that the Pithy instruction data used only 6% of All instruction data

Evaluation of Model Instruction Following Ability After Instruction Tuning

We generated over 1000 test questions separately using ChatGPT and GPT-4. Subsequently, the same test questions were presented to various testing models for responses, including the native model, the model trained by pithy data, and the model trained on the entire dataset. We treated the test questions and answers from multiple testing models as a collective input. We then tasked the evaluation model, GPT-4, with selecting the most suitable answer among those provided by the testing models for each test question. This process is referred to as the winning rate. To mitigate the impact of positional bias, we employed three different evaluation sequences: forward, reverse, and random order. Following the majority principle, we aimed to obtain more accurate results. Positional bias refers to the potential inclination of evaluation models towards answers positioned at specific locations, such as models showing a preference for answers positioned first.

Conclusion

In this paper, we introduce UNIFIT, a unified instruction fine-tuning framework that combines practicality and versatility.UNIFIT makes use of a concept tree to ensure the diversity of instruction data, employs an instruction following difficulty to automatically filter high-quality instruction data(Pithy), and incorporates noise embedding(NE) to promote instruction tuning.UNIFIT has successfully achieved a win rate that exceeds the win rate of using all instruction tweaks by 11% when only 6% of all instruction tweaks are utilized, on top of saving 60% in training time consumption, while ensuring the diversity of instruction tweak data.

The reason for selecting 6% of all instruction data as Pithy data is shown in Fig5. This universal instruction tuning framework signifies a substantial improvement in the practicality and generality of instruction tuning for LLMs. It marks a revolutionary leap in the landscape of instruction tuning, emphasizing that even smaller subsets can yield better instruction-following capabilities, rather than blindly expanding quantity. UNIFIT promises to advancements in efficiency and resource awareness. Unparalleled is that UNIFIT forms a pipeline of instruction data generation, high-quality instruction data selection, and instruction tuning with the addition of noise embeddings, which can be used to propagate these benefits to all LLMs application domains.UNIFIT is actively exploring a broader landing of LLMs to facilitate the improvement of AI in human life and work, especially in terms of efficiency, data quality, and environmental awareness advancement.

Acknowledgments

This work was supported by the National Key R&D Program of China 2023YFB4502100 and the National Natural Science Foundation of China under Grant numbers U2336208

References

Bach, S., Sanh, V., et al. (2022, May). PromptSource: An integrated development environment and repository for natural language prompts. In V. Basile, Z. Kozareva, & S. Stajner (Eds.), *Proceedings of the 60th annual meeting of the association for computational linguistics: System demonstrations* (pp. 93–104). Dublin, Ireland: Association for Computational Linguistics.

Bai, J., Bai, S., et al. (2023). Qwen technical report. *ArXiv*, *abs/2309.16609*.

Cassano, F., Li, L., Sethi, A., Shinn, N., Brennan-Jones, A., Lozhkov, A., … Guha, A. (2023). Can it edit? evaluating the ability of large language models to follow code editing instructions..

Chalnick, A., & Billman, D. (1988). Unsupervised learning of correlational structure. In *Proceedings of the tenth annual conference of the cognitive science society* (pp. 510–516). Hillsdale, NJ: Lawrence Erlbaum Associates.

Chang, Y.-C., Wang, X., et al. (2023). A survey on evaluation of large language models. *ArXiv*, *abs/2307.03109*.

Feigenbaum, E. A. (1963). The simulation of verbal learning behavior. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought.* New York: McGraw-Hill.

Hill, J. A. C. (1983). A computational model of language acquisition in the two-year old. *Cognition and Brain Theory*, *6*, 287–317.

Jain, N., yeh Chiang, P., Wen, Y., Kirchenbauer, J., Chu, H.-M., Somepalli, G., … Goldstein, T. (2023). Neftune: Noisy embeddings improve instruction finetuning. *ArXiv*, *abs/2310.05914*.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de Las Casas, D., … Sayed, W. E. (2023). Mistral 7b. *ArXiv*, *abs/2310.06825*.

Li, M., Zhang, Y., Li, Z., Chen, J., Chen, L., Cheng, N., … Xiao, J. (2023). From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *ArXiv*, *abs/2308.12032*.

Matlock, T. (2001). *How real is fictive motion?* Doctoral dissertation, Psychology Department, University of California, Santa Cruz.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Ohlsson, S., & Langley, P. (1985). *Identifying solution paths in cognitive diagnosis* (Tech. Rep. No. CMU-RI-TR-85-2). Pittsburgh, PA: Carnegie Mellon University, The Robotics Institute.

Shanahan, M., McDonell, K., & Reynolds, L. (2023). Role play with large language models. *Nat.*, *623*(7987), 493–498. Retrieved from `https://doi.org/10.1038/s41586-023-06647-8` doi: 10.1038/S41586-023-06647-8

Shrager, J., & Langley, P. (Eds.). (1990). *Computational models of scientific discovery and theory formation.* San Mateo, CA: Morgan Kaufmann.

Touvron, H., Martin, L., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, *abs/2307.09288*.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., & Hajishirzi, H. (2023, July). Self-instruct: Aligning language models with self-generated instructions. In A. Rogers, J. Boyd-Graber, & N. Okazaki (Eds.), *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 13484–13508). Toronto, Canada: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/2023.acl-long.754` doi: 10.18653/v1/2023.acl-long.754

Wang, Y., Mishra, S., et al. (2022, December). Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In Y. Goldberg, Z. Kozareva, & Y. Zhang (Eds.), *Proceedings of the 2022 conference on empirical methods in natural language processing* (pp. 5085–5109). Abu Dhabi, United Arab Emirates: Association for Computational Linguistics.

Yang, A. M., Xiao, B., et al. (2023). Baichuan 2: Open large-scale language models. *ArXiv*, *abs/2309.10305*.

Zheng, L., Chiang, W.-L., et al. (2023). Judging llm-as-a-judge with mt-bench and chatbot arena. *ArXiv*, *abs/2306.05685*.

Zhou, C., Liu, P., Xu, P., Iyer, S., Sun, J., Mao, Y., … Levy, O. (2023). Lima: Less is more for alignment. *ArXiv*, *abs/2305.11206*.