Examples as the Prompt: A Scalable Approach for Efficient LLM Adaptation in E-Commerce

Jingying Zeng*
zejingyi@amazon.com
Amazon
Palo Alto, CA, USA

Zhenwei Dai*
zwdai@amazon.com
Amazon
Palo Alto, CA, USA

Hui Liu
liunhu@amazon.com
Amazon
Palo Alto, CA, USA

Samarth Varshney
varshsam@amazon.com
Amazon
Palo Alto, CA, USA

Zhiji Liu
liuzhiji@amazon.com
Amazon
Palo Alto, CA, USA

Chen Luo
cheluo@amazon.com
Amazon
Palo Alto, CA, USA

Zhen Li
amzzhn@amazon.com
Amazon
Palo Alto, CA, USA

Qi He
qih@amazon.com
Amazon
Palo Alto, CA, USA

Xianfeng Tang*[✉]
xianft@amazon.com
Amazon
Palo Alto, CA, USA

Abstract

Prompting LLMs offers an efficient way to guide output generation without explicit model training. In the e-commerce domain, prompt-based applications are widely used in query understanding, recommender systems, and customer support. However, adapting LLMs to different tasks often requires extensive prompt engineering by domain experts, along with frequent updates to align with evolving business needs. Additionally, crafting fully unbiased natural language prompts remains a challenge for humans. To address these challenges, we propose a novel framework, <u>E</u>xamples <u>a</u>s the <u>P</u>rompt (EaP). Specifically, EaP automatically selects the most representative examples to maximize the few-shot capability of LLMs. It is efficient due to its unsupervised example selection and adaptive to potential data distribution shifts. We validate EaP on four real-world production use cases, demonstrating that it achieves comparable or even superior performance comparing to hand-crafted prompts designed by domain experts. Additionally, we introduce EaP$_{lite}$, which entirely replaces the natural language components of prompts with labeled examples. EaP$_{lite}$ improves LLM inference speed by up to 70% without compromising performance. The online A/B test shows that using EaP and EaP$_{lite}$ for data labeling can bring significant composite revenue gain by 0.06%.

CCS Concepts

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

Keywords

Large Language Models, Natural Language Generation

---

*Equal contribution. ✉ Corresponding author.

1 Introduction

With the increasing popularity of large language models (LLMs), more and more prompt-based applications are deployed in e-commerce productions such as query understanding [1], recommender system [11], and customer support [15]. Comparing to fine-tuning that requires substantial computational resources and time to update LLM's parameters, prompting enables rapid iteration and adaptation to new customer experiences by simply adjusting the input prompts [2, 14]. This flexibility is particularly advantageous in testing new ideas and minimum viable products, where requirements frequently change. Generally speaking, the main part of a prompt is natural language task definition. Some prompts also include a few in-context examples to illustrate the task [3, 8].

However, designing, improving, and maintaining prompt-based applications still suffer from several limitations: (1) Prompts need regular updates to reflect the latest business needs. Most prompts require extensive hand-crafting by domain experts. To ensure the freshness of the prompts, domain experts need to update the outdated prompt periodically, which is time-consuming and labor-intensive [6]; (2) It can be challenging to incorporate human-annotated data into the prompt. Due to run-time constraints, the length of prompt is limited [12]. Therefore, adding an increasing number of examples to the prompt to cover corner cases is not practical. (3) it is extremely difficult for human to define/describe the business needs with purely unbiased natural languages [4, 20]. For example, when both "very important" and all-caps are used to emphasize constraints, it can be difficult to determine which conveys

greater importance. Moreover, it is challenging to consistently select in-context examples that are both unbiased and sufficiently representative for the application.

Motivated by recent advances in in-context learning (ICL) [2, 18, 21], we aim to address the above challenges by leveraging both LLMs' emerging ability to learn in context, and the rich real-traffic data collected from e-commerce applications. Several studies analyze the importance of in-context example selections to the LLMs' ICL performance [9, 17]. In light of this, instead of improving the natural language instructions in the prompt, ICL leverages labeled data as examples, such that the few-shot capability of LLMs are maximized to perform specified tasks [16]. We propose a novel framework Examples as the Prompt (EaP) that automatically retrieves the optimal in-context examples that cover both corner cases and data distribution shift [19]. EaP constructs the actual prompt on-the-fly with both natural language instructions and dynamic in-context examples selected based on the actual input to the application (e.g., search query). In particular, both global examples representing the task and local examples being similar to the specific input are selected as in-context examples. EaP is unsupervised with no training needed. EaP is flexible and can adopt different retrieval algorithms and similarity measurements. We further introduce EaP_{lite} which completely replaces the system prompts with labeled examples. EaP and EaP_{lite} simplify prompt management and allows us to focus on data annotation quality.

We validate EaP on four e-commerce tasks. EaP consistently shows comparable and even better performance comparing to the original hand-crafted prompt in production. EaP_{lite} significantly reduces the runtime by up to 70%. We test EaP and EaP_{lite} in data labeling pipeline where online A/B test suggests the potential of improvement on composite revenues by 0.06%.

2 Methodology

In e-commerce, LLMs are prompted to generate certain output. For example, in a query parsing task, LLMs take the search query of a user as input, and output the parsing results (e.g., keywords, brand, size, etc.) as json formatted text string. Let q denote the input and o denote the output for the LLMs, we first fit q into a natural language prompt template (i.e., system prompts), then let LLMs read the prompt formatted with q to produce the desired output o.

Because many e-commerce tasks are less common in open-source domain, prompt engineers would hand-pick representative examples for the specific task, and compile them into the system prompt. These in-context examples stimulate the few-shot/zero-shot capabilities of LLMs and improve performances. In most cases, these examples are selected from a large labeled data source accumulated from real business. For example, queries and their human audited/annotated parsing results can be treated as a data source. Figure 1 illustrates the same system prompt without and with in context examples.

Let $\mathcal{E}$ denote the whole labeled data source, EaP aims to find a small set of most representative examples $\mathbf{E} = \{e_1, e_2, \cdots, e_n\} \in \mathcal{E}$ to maximize the few-shot capability of LLMs. In particular, EaP selects examples from both global and local perspectives.
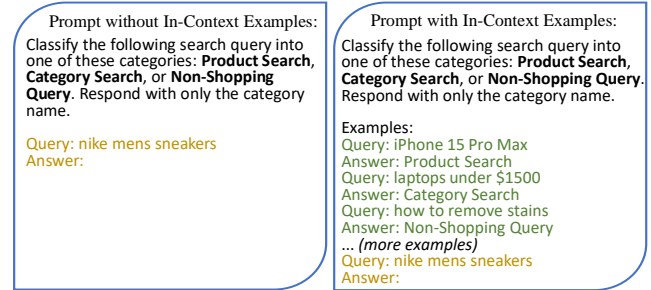

Prompt without In-Context Examples:	Prompt with In-Context Examples:
Classify the following search query into one of these categories: **Product Search**, **Category Search**, or **Non-Shopping Query**. Respond with only the category name. Query: nike mens sneakers Answer:	Classify the following search query into one of these categories: **Product Search**, **Category Search**, or **Non-Shopping Query**. Respond with only the category name. Examples: Query: iPhone 15 Pro Max Answer: Product Search Query: laptops under $1500 Answer: Category Search Query: how to remove stains Answer: Non-Shopping Query ... *(more examples)* Query: nike mens sneakers Answer:

Figure 1: Prompt examples for search query classification. The green text represents in-context examples, while the yellow text denotes the input q.

2.1 Global Examples

Global examples represents the data distribution of the entire problem space for the task. By including global examples in the prompt, we help LLMs understand the problem and learn the correlations between input and output. Ideally, the more representative global examples are chosen, the better performance the prompt can achieve. Global examples can be retrieved through unsupervised methods, such as random selection and K-means clustering. However, when $\mathcal{E}$ is not a uniform sample of the problem space, or when $\mathcal{E}$ contains imbalanced distributed classes, subsampling and/or downsampling techniques can be adopted to reduces the biases in global example selection. When the task is classification, balancing the number of examples from every class can also mitigate the imbalance issues. Because global examples are the same for every input, we can further optimize their representativeness with data condensation approaches [9, 17]. Specifically, a small set $\mathbf{E}^*$ where $|\mathbf{E}^*| = n$ is learned/optimized on $\mathcal{E}$ such that any machine learning model trained on $\mathbf{E}^*$ can achieve the same or very similar performance comparing to the same model trained on $\mathcal{E}$. In practice, replacing manually selected example in the prompt with global examples selected by EaP almost always improves performance. In addition, EaP can update global examples frequently, especially for newly enabled features where user behaviors evolves relatively fast.

2.2 Local Examples

While global examples can already significantly improves the performance, we noticed prompt-based applications could still make mistakes, especially when the input q is less representative. To prevent LLMs from overlooking it, we find examples from $\mathcal{E}$ that are most similar to the specific input, and include them as local examples in $\mathbf{E}$. We first featurize $\mathcal{E}$ with bag of words, TF-IDF, and/or word embeddings, then apply corresponding similarity measurements such as Euclidean distance, Jaccard similarity, and/or cosine similarity. Because local examples are selected in real-time, we further adopt approximate nearest neighbor search methods (e.g., Faiss) to reduce its runtime.

While global examples provide an overview of the problem space, local examples sharp the boundaries between sub-problems and prevent LLMs from being over-confident. Local examples are particularly helpful when q is a corner case or near the boundary of two classes. We illustrate the difference between global and local examples in Figure 2.

Table 1: Comparison of EaP and production prompts Prod.

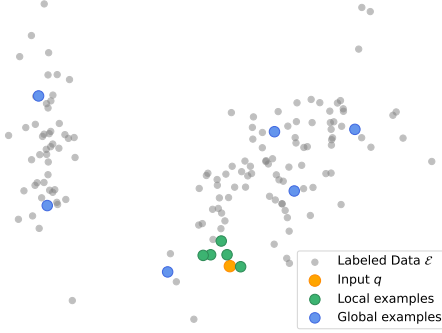| | Navigation | | | | Order | | | | Deal | | Q2Q | |
| | Positive | | Negative | | Positive | | Negative | | | | | |
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	BLEU	ROUGE
Prod	0.8198	0.9040	0.9565	0.9140	0.9916	0.9516	0.9720	0.9952	0.9500	0.9717	0.7736	0.8088
EaP	**0.8910**	**0.9205**	**0.9651**	**0.9513**	**0.9918**	**0.9731**	**0.9843**	0.9952	**0.9715**	**1**	**0.7957**	**0.8136**

Figure 2: Visualization of global and local examples selected by EaP. While global examples represents the overall data distribution, local examples help characterize specific input.

2.3 Performance v.s. Runtime

Comparing to examples in Figure 1, system prompts in real production system can be relatively longer with task definition, formatting instruction, and safety guardrails. Since longer prompts lead to longer inference time for LLMs, the latency constraints limit the total length and total number of in-context examples when deploying prompt-based applications. In practice, we notice that part of the system prompt can be removed when sufficient in-context examples are added by EaP. In most cases, replacing natural language task definition with examples could shorten the prompt and reduce the inference latency for LLMs hosted online.

When latency is not a constraints, adding more examples without modifying existing prompt could achieve the best performance. For offline tasks such as data labeling and generation, we can tune the number of examples in EaP to maximize the performance of prompt-based applications.

3 Experiments

We aim to answer the following three research questions through our experiments: **RQ 1**: Can EaP out-perform state-of-the-art production prompt engineered by human expert? **RQ 2**: How does global and local examples impact the performance of EaP? **RQ 3**: How EaP reduces the inference runtime of LLMs by shorten the prompts?

3.1 Setup

Four real e-commerce tasks with prompt-based solutions are selected to validate EaP:

(1) **Navigation**: In addition to shopping for products, more and more users tend to search for the functionality they need directly in search box. For example, users would search "customer support" directly, rather than navigate to the customer support page. We aim to classify the intent of search queries between navigation and shopping. We collect 730 navigation queries and 1554 regular shopping queries as the labeled.

(2) **Order**: We further want to identify if a search query is clearly asking about a past order placed by the user. We formulate it as a binary classification problem. We collect labels from 1000 past order related queries and 1000 non-order related queries.

(3) **Deal**: Detecting deal related search intent is critical in e-commerce. We collect 2457 search queries with 492 labeled as deal intent.

(4) **Question2Query** (Q2Q): Q2Q transforms question-style search queries into non-question, shoppable formats. For example, it converts "how to clean a cast iron skillet" into "cast iron skillet cleaning." Q2Q enhances the coverage and effectiveness of traditional e-commerce search functions, such as rule-based solutions, before transitioning to modern machine learning and LLM-based approaches. We collect 1543 question-style queries paired with their corresponding shoppable counterparts.

We randomly split the collected labeled data equally into validation set and test set. We use validation set as the labeled data $\mathcal{E}$, and compare EaP against our state-of-the-art human engineered prompts in production. Precision and Recall metrics are reported on Navigation, Order and Deal; while BLEU [13] and ROUGE [10] are reported for Q2Q.

Without the loss of generality, we adopt Mistral-7B-Instruct-v0.2[1] [5] with vLLM [7] for our experiment. To ensure a fair comparison, the total number of examples selected by EaP are always equal to the number of examples in production prompts on every task. We use sklearn K-means clustering[2] to select global examples, and use RapidFuzz[3] to find local examples in our offline evaluation. All experiments are conducted on a single AWS EC2 P4 Instances with A100 GPUs. Note that GPU is only used for LLM inference and example selection in EaP are conducted on CPU only.

3.2 Performance Analysis

We report the performance comparison of EaP and Prod in Table 1. Based on the results, we observe that EaP can consistently outperform the handcrafted production prompts over all the tasks, which shows the advantages of smartly selecting few-shot examples in ICL over static human-written prompt. EaP achieves significant improvements on detecting navigational intent in the task Navigation. One potential reason is that Navigation is hard to be defined precisely via natural language definition. Leveraging representative global and local examples from EaP could mitigate the gap in system prompt and improve the performance. For other tasks such as Order, Deal, and Q2Q, EaP achieves slightly better performance than the Prod prompt. However, EaP shortens the prompt lengths by eliminating the length natural language instructions from system prompt, which can significantly reduce the latency during online service. In Section 3.3, we will have a more thorough analysis on the inference speed comparison.

---

[1]https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2
[2]https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
[3]https://github.com/rapidfuzz/RapidFuzz

3.3 Global and Local Examples

For a fixed budget of total number of examples, we investigate how to balance the number of global and local examples to maximize the performance. Taking Navigation as an example, we keep the total number of in-context examples unchanged, while alter the number of global and local examples. The prompt used in production includes 11 in-context examples. To have a fair comparison to the production setup, we also set the total number of in-context examples to 11. The precision and recall on both positive and negative class for navigational intent classification are reported in Table 2. We notice the both precision and recall will first increase and then decrease, showing that balanced numbers of global and local examples are important for EaP. Moreover, we observe that the positive class is more sensitive to the change, due to the fact that it is the minority class in this unbalanced dataset. Selecting the right balance of global and local examples is crucial to EaP performance.

Table 2: Changing the number of global and local examples.

# global examples	# local examples	Positive Precision	Recall	Negative Precision	Recall
11	0	0.8653	0.8264	0.9232	0.9419
10	1	0.8134	0.8971	0.9213	0.9071
9	2	0.8066	0.8714	0.9398	0.9057
8	3	0.8613	0.8585	0.9362	0.9376
7	4	0.8696	0.8360	0.9273	0.9434
6	5	0.8850	0.8907	0.9505	0.9478
5	6	**0.9010**	**0.9205**	**0.9651**	0.9513
4	7	0.9066	0.8424	0.9311	**0.9608**
3	8	0.8822	0.8907	0.9504	0.9463
2	9	0.8300	0.8778	0.9648	0.9144
1	10	0.8112	0.8842	0.9455	0.9071
0	11	0.7881	0.8971	0.9505	0.8912

3.4 Effectiveness of System Prompt

For the classification tasks like Navigation, LLM heavily relies on the in-context examples to make the prediction, while the system prompt can only provide general and high-level guidance. In this experiment, we investigate to which extend the role of system prompt can be replaced by the in-context examples. Surprisingly, we found that even when removing the system prompt and replacing it with a simple one-line instruction "***Here are some examples of expected input and output:***" The LLM still achieves similar performance, provided sufficient in-context examples. We create an ablation EaP$_{lite}$ with such one-line system prompt and study how its performance would change w.r.t the number of examples on Task Navigation. We fix the ratio of global and local examples according to the learning in 3.3. We compare the inference speed of each method based on the following two two metrics: (1) **Item per second (it/s)**: this metrics quantifies the number of prompts that can be processed per second. (2) **Output token per second (toks/s)**: this metrics quantifies how many output tokens are generated per second. We report the overall accuracy (i.e., if a query intent is correctly classified) in Table 3. The performance of EaP converges after the total number of examples reaches about 11. Even though the inference speed decreases almost linearly with the addition of more examples, EaP still runs faster than the hand-crafted prompt Prod and EaP, even when 39 examples are used. This motivate us to supress and/or remove system prompt when possible in production.

Table 3: EaP$_{lite}$ outperforms Prod with no system prompt.

# global examples	# local examples	Total # samples	Accuracy	Input it/s	Output toks/s
1	2	3	0.819	48.11	3,079.31
2	3	5	0.884	45.71	2,925.23
3	4	7	0.911	44.07	2,820.79
4	5	9	0.928	41.90	2,681.76
5	6	11	0.932	40.11	2,566.81
6	7	13	0.933	38.93	2,491.43
7	8	15	0.938	37.40	2,393.90
8	9	17	0.948	35.72	2,286.09
9	10	19	0.941	34.28	2,194.15
10	11	21	0.947	32.07	2,052.43
11	12	23	0.942	31.90	2,041.87
12	13	25	0.942	30.65	1,961.50
13	14	27	0.957	29.87	1,911.49
14	15	29	0.957	28.57	1,828.71
15	16	31	0.942	27.80	1,779.30
16	17	33	0.943	26.84	1,717.46
17	18	35	0.948	26.37	1,687.84
18	19	37	0.950	25.58	1,636.85
19	20	39	0.944	25.25	1,616.19
Prod		11	0.910	23.16	1,482.20
EaP (5 global + 6 local)		11	0.932	23.16	1,482.20

4 Production Usecases

EaP can be widely adopted to improve data labeling and assist human auditing. With a small set of high-quality human annotated data, we can leverage EaP to quickly expand them to a much larger LLM-labeled dataset. We report the efficiency improvement by adopting EaP$_{lite}$ into the data labeling pipelines for task Navigation, Order, and Deal in Table 4 using the same offline datasets. Clearly, EaP$_{lite}$ significantly increases the output token per second speed by up to 70%, due to the shorter prompts.

Table 4: EaP$_{lite}$ imporves data labeling efficiency.

	Navigation it/s	toks/s	Order it/s	toks/s	Deal it/s	toks/s
Prod	23.16	1482.20	27.19	1740.45	25.36	1623.32
EaP$_{lite}$	39.74	2543.59	44.42	2842.74	41.3	2643.44
Improvement	71.59%	71.61%	63.37%	63.33%	62.85%	62.84%

Downstream applications can utilize extra data labeled by EaP for their model training. One example is expanding labeled dataset size for underrepresented classes. For example, recognizing new brands is relatively more challenging than their well-established counterpart. We leverage EaP to expanding the labeled data for Brand Named entity recognition (NER) model. Based on large-scale online A/B test, the model trained on expanded labeled data shows significant top-line metrics gains on composite revenues by 0.06% .

5 Conclusion

This paper introduces a novel approach, EaP, to select representative in-context learning examples that can optimize the LLM prediction/generation quality. Compared to static, human-crafted prompts, EaP eliminates the need for costly human effort while achieving superior performance. Further analysis reveals that the example-only prompt, which replaces the human-designed system prompt with a simple one-line instruction, maintains comparable performance while significantly reducing latency. We envision that our approach can be implemented across various domains, fostering faster development of LLM applications in industry.

References

[1] Avishek Anand, Abhijit Anand, Vinay Setty, et al. 2023. Query understanding in the age of large language models. *arXiv preprint arXiv:2306.16004* (2023).

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[3] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234* (2022).

[4] Seraphina Goldfarb-Tarrant, Eddie Ungless, Esma Balkir, and Su Lin Blodgett. 2023. This prompt is measuring< mask>: evaluating bias evaluation in language models. *arXiv preprint arXiv:2305.12757* (2023).

[5] Albert Q Jiang, A Sablayrolles, A Mensch, C Bamford, D Singh Chaplot, Ddl Casas, F Bressand, G Lengyel, G Lample, L Saulnier, et al. 2023. Mistral 7b. arxiv. *arXiv preprint arXiv:2310.06825* 10 (2023).

[6] Weize Kong, Spurthi Amba Hombaiah, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Prewrite: Prompt rewriting with reinforcement learning. *arXiv preprint arXiv:2401.08189* (2024).

[7] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

[8] Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060* (2024).

[9] Zhuowei Li, Zihao Xu, Ligong Han, Yunhe Gao, Song Wen, Di Liu, Hao Wang, and Dimitris N Metaxas. 2024. Implicit In-context Learning. *arXiv preprint arXiv:2405.14660* (2024).

[10] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[11] Han Liu, Xianfeng Tang, Tianlang Chen, Jiapeng Liu, Indu Indu, Henry Peng Zou, Peng Dai, Roberto Fernandez Galan, Michael D Porter, Dongmei Jia, et al. 2024. Sequential LLM framework for fashion recommendation. *arXiv preprint arXiv:2410.11327* (2024).

[12] Qibang Liu, Wenzhe Wang, and Jeffrey Willard. 2025. Effects of Prompt Length on Domain-specific Tasks for Large Language Models. *arXiv preprint arXiv:2502.14255* (2025).

[13] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 311–318.

[14] Jiho Shin, Clark Tang, Tahmineh Mohati, Maleknaz Nayebi, Song Wang, and Hadi Hemmati. 2023. Prompt engineering or fine tuning: An empirical assessment of large language models in automated software engineering tasks. *arXiv preprint arXiv:2310.10508* (2023).

[15] Hanchen Su, Wei Luo, Yashar Mehdad, Wei Han, Elaine Liu, Wayne Zhang, Mia Zhao, and Joy Zhang. 2025. LLM-Friendly Knowledge Representation for Customer Support. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*. 496–504.

[16] L. Wang, N. Yang, and F. Wei. 2023. Learning to retrieve in-context examples for large language models. *arXiv preprint* (2023). arXiv:2307.07164

[17] Xinyi Wang, Wanrong Zhu, and William Yang Wang. 2023. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. *arXiv preprint arXiv:2301.11916* (2023), 3.

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

[19] J. Wei, Y. Tay, D. Tran, A. Webson, Y. Lu, X. Chen, H. Liu, D. Huang, D. Zhou, and T. Ma. 2023. Larger language models do in-context learning differently. *arXiv preprint* (2023). arXiv:2303.03846

[20] Ziyang Xu, Keqin Peng, Liang Ding, Dacheng Tao, and Xiliang Lu. 2024. Take care of your prompt bias! investigating and mitigating prompt bias in factual knowledge extraction. *arXiv preprint arXiv:2403.09963* (2024).

[21] Yuxiang Zhou, Jiazheng Li, Yanzheng Xiang, Hanqi Yan, Lin Gui, and Yulan He. 2023. The mystery of in-context learning: A comprehensive survey on interpretation and analysis. *arXiv preprint arXiv:2311.00237* (2023).