Reducing Model Sensitivity via Single-Elimination Tournament Inference

Anonymous ACL submission

Abstract

We present SEDE, a simple method that solves item selection problems (e.g., multiple choice question answering) via single-elimination tournaments. Specifically, SEDE decomposes an initial selection problem into a collection of *smaller* problems; each having fewer items than the initial problem. By virtue of being short and simple, state-of-the-art language models avoid the performance degradation that arises from position bias and long inputs, and thus solve each small problem accurately. In experiments on 3 tasks—multiple choice question answering, multi-document question answering, and software issue localization—we show that SEDE leads to higher accuracy than both an in-context learning and a position debiased baseline on selection problems with many items. Unlike the baselines, SEDE is robust to increases in the number of items. Our analysis reveals that improvements due to SEDE over the other two methods are more dramatic as model size decreases and problem size increases.

1 Introduction

In-context learning (ICL) is perhaps the most dominant paradigm for accomplishing tasks with large language models (LLMs). To complete a task via ICL, a practitioner employs a pre-trained model, and supplies it with an input that typically contains: i) natural language *instructions* describing the task, ii) a collection of sample task inputs-output pairs, termed *demonstrations*, and iii) a task input with no corresponding output, known as the *prompt* (Brown et al., 2020). Ideally, the LLM produces the corresponding output via autoregressive text generation.

As a mode of model interaction, ICL has many advantages. Unlike finetuning, ICL does not require gathering labeled datasets for training. Moreover, ICL facilitates successful task completion with state-of-the-art, close-source models via APIs. Indeed, many state-of-the-art results are accomplished using ICL rather than finetuning, or other forms of training. For example, Medprompt with GPT-4 yields the best performance on a suite of medical question answering tasks and agent-computer interfaces lead to the most performant approaches for automated software engineering (Yang et al., 2024; Nori et al., 2023).

However, under the ICL paradigm, LLMs are sensitive to the composition of their input (Dong et al., 2022). In particular, LLMs are sensitive to input order. For instance, recent work shows that in multi-document question answering (MultiQA), the likelihood that an LLM answers a question correctly is significantly impacted by the location of the passage containing the answer in the input (Liu et al., 2024a). Similarly, when employing ICL for multiple choice question answering (MCQA), the position of the correct answer among the candidate options strongly affects model correctness—a phenomenon known as *position bias*. This has been demonstrated using the most performant models, such as GPT-4 (Zheng et al., 2023a,b; Wang et al., 2023; Pezeshkpour and Hruschka, 2023). Varying the order of the demonstrations in the input has also been shown to cause performance to fluctuate between state-of-the-art and random chance (Lu et al., 2022; Zhao et al., 2021).

Another issue that manifests under the ICL paradigm, is that LLMs tend to perform worse as the length of their input increases. For example, in MultiQA, model performance drops substantially as the number of documents in the prompt increases (Liu et al., 2024a). This result holds for models trained specifically to utilize long contexts. A similar finding is reported in extreme classification (i.e., classification with a large number of classes): as the number of tokens in the task prompt increases, classification accuracy decreases (Li et al., 2024). While this seems at odds with the notion that adding task demonstrations im-

proves accuracy, recent work shows that too many demonstrations incurs performance drops, and does not lead to variance reduction (Zhao et al., 2021).

To simultaneously decrease sensitivity to position bias and to length, we study a simple decomposition of long inputs into a collection of smaller sub-problems. Specifically, we propose SEDE, which solves item selection problems—like MCQA—using single-elimination tournament. In more detail, given an item selection problem, SEDE iteratively builds *contests*, i.e., variants of the initial problem that have a small subset of its items. For each contest, SEDE employs an LLM for selection. New contests are formed using the selected items; non-selcted items are eliminated. The process continues until a single item remains, which is returned as the final answer.

We experiment with SEDE on 3 tasks: MCQA, MultiQA, and software issue localization (IssLoc). While SEDE can be deployed directly on MCQA, for MultiQA and IssLoc, we utilize simple transformations that convert instances of these problems into MCQA. On 5 datasets across these tasks, we observe that SEDE achieves high accuracy and low variance. In comparison to both an ICL and a position debiased baseline, SEDE is always the best or among the best methods. We find that performance improvements due to SEDE are most significant on problems with the longest inputs. Moreover, this result holds across all models in the Llama family, with the smallest models enjoying the most dramatic improvements. By virtue of building small contests with few items, SEDE helps to alleviate the issues stemming from position bias and long inputs.

2 Single-Elimination Tournament Inference

Our goal is to reduce the sensitivity of LLMs to input length and input order under the ICL paradigm. To do so, we propose a simple inference method—**s**ingle **e**limination **de**composition (SEDE, pronounced "seed")—that decomposes a problem with a long input into a collection of problems with shorter inputs. Before detailing our method, we briefly define notation.

2.1 Notation

Let $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ be a model that takes an input $x \in \mathcal{X}$ and produces an output $y \in \mathcal{Y}$. Without loss of generality, let x be an instance of MCQA having $k > 1$ *candidate* options, i.e., $\mathsf{cand}(x) = \{c_0, \cdots, c_{k-1}\} \subset \mathcal{Y}$, where $\mathsf{cand}(.)$ returns the candidates of its argument. We say that x has *size* k if $|\mathsf{cand}(x)| = k$.

2.2 Single-Elimination Tournament Inference

SEDE solves MCQA problem x, by executing a *single-elimination tournament* among its candidates, $\mathsf{cand}(x)$. Formally, let T be a single-elimination tournament, which proceeds in rounds. During each round, SEDE eliminates some candidates in $\mathsf{cand}(x)$ from consideration. After the final round, only 1 candidate remains, which is returned as the solution of x.

In detail, in the first round of the tournament, T_0, none of $\mathsf{cand}(x)$ have been eliminated. For that round, we form a collection of *contests*, where a contest, $\hat{x}$, is identical to x except that $\mathsf{cand}(\hat{x}) \subset \mathsf{cand}(x)$, i.e., the candidates of $\hat{x}$ are a subset of the candidates of x. While the candidates for a specific contest may be chosen deliberately, in general, we assume that they are sampled from the non-eliminated candidates. For any pair of contests in round i, $\hat{x}_{ij}, \hat{x}_{ik}$, the corresponding candidates sets are disjoint, i.e., $\mathsf{cand}(\hat{x}_{ij}) \cap \mathsf{cand}(\hat{x}_{ik}) = \varnothing$. The model, $\mathcal{M}$, is employed to select the *winner* of each contest, $\mathcal{M}(\hat{x}_{ij}) = w_{ij}$; all non-winning candidates are eliminated. The contests comprising round $i + 1$ are formed from the winners of the contests of round i. The tournament ends when a single candidate remains.

2.3 Computational Cost

Consider running SEDE on input x and let $|\mathsf{cand}(x)| = k$. Assuming that all contests in all rounds are of size, b, then the total number of contests is $\sum_{i=0}^{\log_b(k)-1} b^i$. As such, if round i yields W winners, then round $i + 1$ will contains $\lfloor \frac{W}{b} \rfloor$ contests of size b, and at most 1 contest of size $W - \lfloor \frac{W}{b} \rfloor$. For any contest of size one, the corresponding candidate is considered to be a winner, and moves on to the next round.

3 Experiments

In this section, we report the performance of SEDE on 3 tasks: multiple choice question answering (MCQA), Multi-document QA (MultiQA), and software issue localization (IssLoc). We experiment with the family of Llama2 models, specifically the 7, 13, and 70 billion parameter variants.

We measure the accuracy and variance of SEDE, and compare it to both standard ICL and a position debiased variant of ICL (DEBIAS). DEBIAS is implemented by sampling p random permutations of the problem candidates, $\mathsf{cand}(x)$, conducting standard ICL for each permutation, and returning the majority vote; ties are resolved arbitrarily. Importantly, we set $p = \sum_{i=0}^{\log_b(k)-1} b^i$, i.e., the number of contests run by SEDE, so that SEDE and DEBIAS incur the same number of calls to the LLM.

3.1 Multiple Choice QA

We experiment with 3 MCQA datasets: Commonsense QA (**CSQA**), **ARC-Easy**, and **ARC-Challenge** (Talmor et al., 2019; Clark et al., 2018). For each dataset, we study the sensitivity of each method to instance size (i.e., the number of candidate options). To alter the number of candidates, we randomly remove and/or add *incorrect* candidates. For a problem instance, x, we add incorrect candidate options by randomly sampling from the candidate options of *other* instances in the training set, i.e., $\bigcup_{x' \in \mathcal{X}_{train} \setminus x} \mathsf{cand}(x')$.

For each method, we supply k ICL demonstrations, where k is the size of the problem before we add or remove candidates[1]. We evaluate each method on each instance 10 times. For each repetition, for all methods, we sample a new set of demonstrations from among the training set instances. For SEDE, in each repetition, we also sample new contests among the candidate options.

For the 3 datasets, Figure 1 visualizes the mean accuracy, as well as the minimum and maximum accuracy with respect to experiment repetitions, versus instance size for the `Llama2-7B` model. The Figure reveals that for problem sizes of $\sim$15 or fewer, SEDE and DEBIAS achieve similar accuracy and both outperform ICL. However, as problem size increases, SEDE dominates both methods. Remarkably, SEDE experiences very slow degradation in accuracy and little variation over experimental repetitions, underscoring the robustness of the approach. These trends consistently appear in larger models, albeit less dramatically refapp:results.

[1]We do this in order to supply a demonstration of the correct answer being in each position. We do not do this for larger values of k since, for large problem sizes, this ultimately exhausts the context length.

3.2 Multi-document QA & Issue Localization

Next, we test the suitability of SEDE in two additional tasks: multi-document QA (MultiQA) and software issue localization (IssLoc). In both, we reduce the initial problems to instances of MCQA and apply SEDE.

3.2.1 MultiQA

For MultiQA, we study the Natural Questions dataset (**NQ**) (Kwiatkowski et al., 2019). In the task, the input is a question and a collection of documents, one of which—the gold document—that contains the answer to the question. The goal is to answer the question correctly. We utilize the questions and retrieved passages used in recent work (Liu et al., 2024a). Like their work, we study variants of the dataset containing either 10, 20 or 30 candidate documents per question (respectively, **NQ-10**, **NQ-20**, and **NQ-30**). Rather than directly answering the questions, we study the task of correctly identifying the gold document, which reduces the problem to MCQA.

3.2.2 Software Issue Localization

We derive a software issue localization dataset from SWE-Bench—a software engineering benchmark containing GitHub issues and the corresponding *patches*, i.e., code revisions (Jimenez et al., 2024). In our dataset, **SWE-MC**, the input is a GitHub issue—like in SWE-Bench—as well as a collection of candidate code chunks (e.g., function implementations), and the goal is to identify the chunk that needs to be patched. As such, our dataset is an instance of MCQA.

Dataset Construction. In detail, in **SWE-MC**, the input is a natural language GitHub issue, and the corresponding candidates are sequences of contiguous lines of code. All candidates are disjoint, i.e., no line of code appears in more than 1 candidate, and are extracted from the same file. To form candidates, we start by building the abstract syntax tree (AST) of the file to be patched. From the AST, we extract all classes and functions. For any line of code, l, that line belongs a candidate including: i) all lines before l until the nearest function or class declaration, and ii) all lines after l until the nearest function or class boundary. In both cases, we measure closeness in terms of lines of code. To simplify evaluation, we only use SWE-Bench instances for which the corresponding patch is a single, contiguous chunk of code.

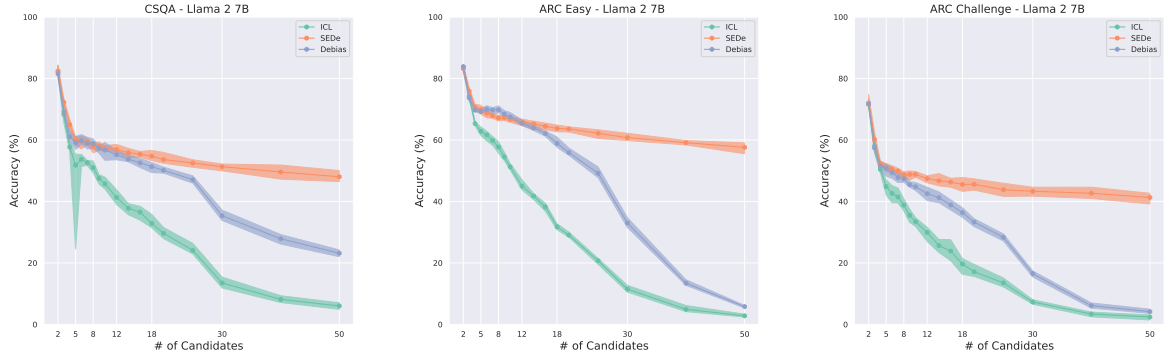

Figure 1: **Accuracy and Variation in MCQA.** The mean accuracy on 3 MCQA datasets when the number of candidate options are varied. Errors bars indicate minimum and maximum accuracy over 10 trials.

In other words, we omit instances whose patches span multiple files and/or multiple functions. Similar filters have been employed to construct other software engineering benchmarks (Muennighoff et al., 2024). We discard all SWE-Bench issues that come from non-permissively licensed repositories[2]. For each SWE-MC instance, we select candidates in ascending order of tokens, and form 3 variants of dataset: **SWE-MC-10**, **SWE-MC-20**, and **SWE-MC-30** with 10, 20, and 30 candidates per question, respectively. This leaves us with 114 instances of issue localization.

3.3 Results

Instances in **NQ** and **SWE-MC** are long (in terms of number of tokens), and so we utilize either 0 or 1 demonstration per problem for each method. Since SEDE constructs smaller problem instances, we also evaluate it with 5 ICL examples per dataset; a setting that exceeds the context length for the other methods. Unlike the previous experiments, we do not conduct multiple repetitions.

Results for `Llama2-7B` are reported in Table 1. Overall, we observe that SEDE dominates the other methods. This is especially the case in the longer variants of both datasets. When the number if ICL examples among the methods is equal, their respective accuracy of each method is closer. However, when we increase the number of ICL examples for SEDE to 5—which is enabled specifically by its problem decomposition—SEDE improves on the accuracy of the next closest competitor by 56.4%, absolutely. As model size increases, each method improves, but SEDE remains the top performer.

[2]We only use issues from repositories with an MIT, Apache-2.0, or BSD-3 license.

Method	# Demos	NQ-10	NQ-20	NQ-30
SEDE	0	*0.1*	*0.08*	*0.08*
DEBIAS	0	0.04	0.02	0.02
ICL	0	0.04	0.02	0.02
SEDE	1	*0.39*	*0.37*	*0.37*
DEBIAS	1	0.15	—	—
ICL	1	0.13	—	—
SEDE	5	**0.72**	**0.78**	**0.78**

Table 1: **MultiQA Accuracy: `Llama2-70B`.** Dashed (—) entries indicate that the maximum token limit is exceeded when the corresponding number of ICL demonstrations are utlized. *Italic* entries are the best performers given a fixed number of ICL demonstrations; **bolded** entries indicate best overall.

4 Related Work

A flurry of recent work is focused on reducing the impact of position bias, especially for MCQA. Among the methods that have been proposed, many suggest reordering the options multiple times to effectively marginalize out the impact of order (Wang et al., 2023; Zheng et al., 2023b,a). Similarly, in our experiments, we compare to a position debiased baseline—DEBIAS—which computes the majority vote among model generations over multiple shuffles of the candidate order. One piece in this space also highlights *token bias*, i.e., that certain multiple choice answer labels—like "A"—have higher likelihood a priori (Zheng et al., 2023a). While we do not specifically account for token bias, we believe that our DEBIAS baseline largely mitigates its effect. Related, yet more distant from our work, are studies focused on mitigating position bias in tasks beyond MCQA (Liu et al., 2024b; Shinoda et al., 2022).

Whereas we attack issues arising from long inputs by breaking the input down into smaller parts, most work in this space focuses on enabling models to better handle long context. For example, one family of methods develops approaches that sparsify or compress the attention computation (Child et al., 2019; Zaheer et al., 2020; Beltagy et al., 2020; Roy et al., 2021). A recent trend is to simply train large models with increasingly large limits on maximum token inpus. We do not experiment with these "extended-context models" since they do not confer advantages over standard models when the input length is within standard token limits (Liu et al., 2024a). We note that in our experiments, we ensure that no input ever exceeds the maximum token length for the model employed. Moreover, while we increase input lengths by adding additional items from which to choose, in many cases, the number of tokens in an input remains well below the corresponding model's threshold. For more information on efficient transformer models, we refer the interested reader to a survey on the topic (Tay et al., 2022).

Our approach draws inspiration from classic work on reducing multi-class classification to a sequence of binary decisions (Allwein et al., 2000). Among the most similar to our work is the Filter Tree, which solves multi-class classification by learning a tree of classifiers that represents a single elimination tournament (Beygelzimer et al., 2009). Unlike the Filter Tree, we utilize ICL with one LLM rather than training a separate model at every split of the tree. Whereas the binary constests in the Filter Tree are fixed, we build contests by randomly sample the candidates of the (initial) problem instance. Finally, classic approaches, like the Filter Tree, are only capable of handling a fixed set of classes, while SEDE is focused on the more general class of tasks that can be converted to MCQA.

5 Conclusion

Since LLMs have been shown to be sensitive to input length and position bias, especially in the ICL paradigm, we propose SEDE, a simple decomposition of long, multiple choice problems into single-elimination tournaments. Aside from traditional multiple choice tasks, we study multi-document question answering and software issue localization, by converting datasets for each task into instances of MCQA. In experiments, we show that SEDE achieve higher accuracy than both ICL and DEBIAS—a position debiased variant of ICL—especially for problem instances with many candidates and for smaller LLMs. As long as LLMs continue to be sensitive to their inputs in the ICL paradigm, we believe that SEDE will prove useful in improving accuracy on item selection tasks.

6 Limitations

Our experiments only study the `Llama2` family of models. This is because, at the time of experimentation, these represented the most performant open source models available. We note that previous work, which we discuss, experiment with stronger closed source models—namely, GPT-4—and find that position bias and negative correlations with length exist. As such, we believe that our results would extend to stronger models.

SEDE's utility is limited to decomposable selection problems. We demonstrate that these selection problems go beyond traditional multiple choice tasks. Specifically, in experiments we show that some instances of multi-document question answering and software issue localization can be decomposed into single-elimination tournaments. However, some tasks require long inputs, and others—like multi-hop QA— may require multiple selections, which may pose additional challenges if employing SEDE (Shaham et al., 2022; Qi et al., 2021; Yang et al., 2018).

SEDE side-steps the issue of long context understanding for LLMs by representing long inputs as a collection of small problems. From a philosophical perspective, training better models that do not exhibit these issues is the ultimate solution. Indeed, it is claimed that the Gemini 1.5 Pro model solves the task of finding small facts in up to 1M tokens with 99% accuracy (i.e., the needle in a haystack test) (gem). However, until the point where this type of performance is available among open source models, SEDE provides a conceptually simple and easy-to-implement mechanism for boosting accuracy and reducing variance for a wide collection of tasks.

References

MS Windows NT kernel description. https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/. Accessed: 2024-06-11.

Erin L Allwein, Robert E Schapire, and Yoram Singer. 2000. Reducing multiclass to binary: A unifying

approach for margin classifiers. *Journal of machine learning research*, 1(Dec):113–141.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Alina Beygelzimer, John Langford, and Pradeep Ravikumar. 2009. Error-correcting tournaments. In *International Conference on Algorithmic Learning Theory*, pages 247–262. Springer.

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*, 33:1877–1901.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

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*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Zhongkun Liu, Zheng Chen, Mengqi Zhang, Zhaochun Ren, Zhumin Chen, and Pengjie Ren. 2024b. Zero-shot position debiasing for large language models. *arXiv preprint arXiv:2401.01218*.

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, Dublin, Ireland. Association for Computational Linguistics.

Niklas Muennighoff, Qian Liu, Armel Randy Zebaze, Qinkai Zheng, Binyuan Hui, Terry Yue Zhuo, Swayam Singh, Xiangru Tang, Leandro Von Werra, and Shayne Longpre. 2024. Octopack: Instruction tuning code large language models. In *The Twelfth International Conference on Learning Representations*.

Harsha Nori, Yin Tat Lee, Sheng Zhang, Dean Carignan, Richard Edgar, Nicolo Fusi, Nicholas King, Jonathan Larson, Yuanzhi Li, Weishung Liu, et al. 2023. Can generalist foundation models outcompete special-purpose tuning? case study in medicine. *arXiv preprint arXiv:2311.16452*.

Pouya Pezeshkpour and Estevam Hruschka. 2023. Large language models sensitivity to the order of options in multiple-choice questions. *arXiv preprint arXiv:2308.11483*.

Peng Qi, Haejun Lee, Oghenetegiri "TG" Sido, and Christopher D. Manning. 2021. Answering open-domain questions of varying reasoning steps from text. In *Empirical Methods for Natural Language Processing (EMNLP)*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Uri Shaham, Elad Segal, Maor Ivgi, Avia Efrat, Ori Yoran, Adi Haviv, Ankit Gupta, Wenhan Xiong, Mor Geva, Jonathan Berant, and Omer Levy. 2022. SCROLLS: Standardized CompaRison over long language sequences. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 12007–12021, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kazutoshi Shinoda, Saku Sugawara, and Akiko Aizawa. 2022. Look to the right: Mitigating relative position bias in extractive question answering. In *Proceedings of the Fifth BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 418–425.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*,

pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. 2024. Swe-agent: Agent-computer interfaces enable automated software engineering. *arXiv preprint arXiv:2405.15793*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

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.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2023a. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023b. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

A Results

In this section, we report the results elided in Section 3. Specifically, we visualize MCQA performance for `Llama2-13B` and `Llama2-70B` in Figures 2 and 3, respectively. We do not conduct multiple repetitions of each experiment as before in order to reduce computational cost. Given the low variation of SEDE in the previous experiments, we have reason to believe that these results are similarly robust.

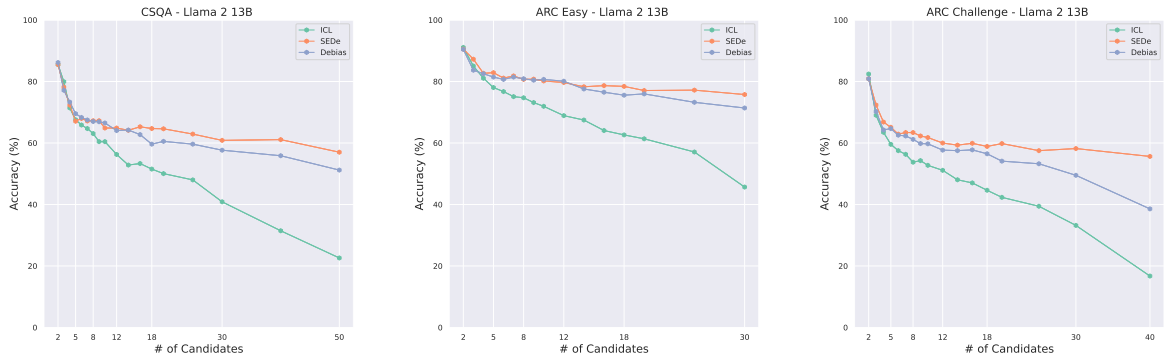

Figure 2: **MCQA Accuracy: `Llama2-13B`.** MCQA accuracy on 3 datasets as the number of candidate options are varied.

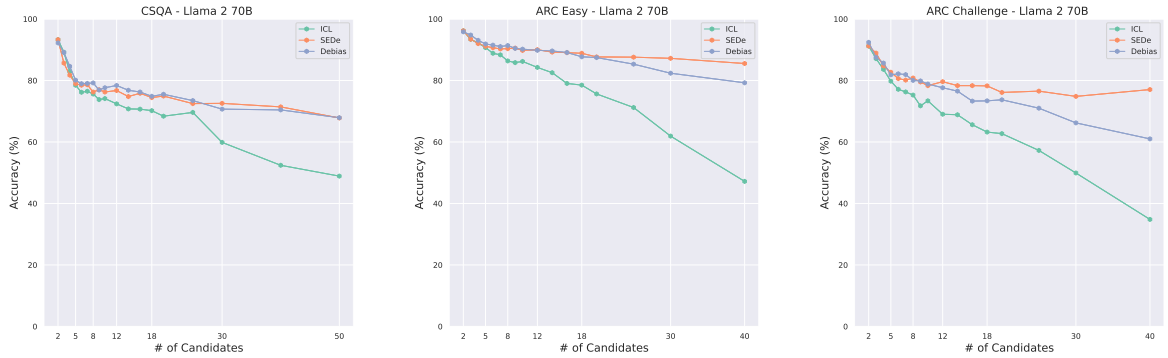

Figure 3: **MCQA Accuracy: `Llama2-70B`.** MCQA accuracy on 3 datasets as the number of candidate options are varied.