Verbalized Algorithms

Supriya Lall
MIT CSAIL
MIT-IBM Watson AI Lab
`supriya@mit.edu`

Christian Farrell
Marist University
IBM Infrastructure
`christian.farrell@ibm.com`

Hari Pathanjaly
UC Irvine
IBM Infrastructure
`hari.pathanjaly@ibm.com`

Marko Pavic
Marist University
IBM Infrastructure
`marko.pavic@ibm.com`

Sarvesh Chezhian
IBM Infrastructure
`schez@ibm.com`

Masataro Asai
MIT-IBM Watson AI Lab
IBM Research Cambridge, USA
`masataro.asai@ibm.com`

Abstract

Instead of querying LLMs in a one-shot manner and hoping to get the right answer for a reasoning task, we propose a paradigm we call *verbalized algorithms* (VAs), which leverage classical algorithms with established theoretical understanding. VAs decompose a task into simple elementary operations on natural language strings that they should be able to answer reliably, and limit the scope of LLMs to only those simple tasks. For example, for sorting a series of natural language strings, *verbalized sorting* uses an LLM as a binary comparison oracle in a known and well-analyzed sorting algorithm (e.g., bitonic sorting network). We demonstrate the effectiveness of this approach on sorting and clustering tasks.

1 Introduction

Imagine receiving thousands of customer complaints and having to quickly sort them by how urgently they need support. Feeding such a large set of input strings blindly into a large language model (LLM) is ineffective and impractical, both due to its limited context length and its limited ability to perform sorting tasks. LLMs provide no guarantees on the correctness of their outputs for computational reasoning tasks, such as sorting, beyond anecdotal, empirical observations that they *sometimes* do. Despite the recent improvements, their behavior is fundamentally an approximation of formal reasoning, and thus contains hallucinations.

In contrast, traditional disciplines of computer science often approach such a task using task decomposition. For example, a merge sort decomposes the entire sorting task into several sorting tasks over individual sublists and combines the results, where, at the limit, there is an elementary binary comparison operator that compares individual pairs of objects in the list. These traditional algorithms, given an accurate oracle, guarantee the correctness of their outputs and the average/worst-case runtime.

Such an algorithmic and analytical approach is often missing in LLM-based reasoning literature due to opaqueness of the reasoning mechanism in LLMs. More precisely, it often fails to distinguish the various separate challenges of reasoning tasks, i.e., (1) the ability to accurately perform atomic operations on ambiguous representations, (2) the ability to accurately execute a (not necessarily efficient) algorithm using those operations, (3) the choice of the algorithm to run and its theoretical properties (runtime/memory complexity, parallelism, robustness, soundness, completeness, etc.), and (4) the heuristic policy/hints for a branching algorithm. We argue that most reasoning tasks are informal representations of formal tasks for which *an ideal classical algorithm with known complexity guarantees already exists in the literature*, thus the effort to further improve (2,3) via additional training is wasteful and unproductive. For example, for the comparison sort, $O(\log n)$ algorithms are asymptotically optimal.

In this work, we propose an abstract framework called "verbalized algorithms". Given an informal representation of a formal task, a verbalized algorithm operates by replacing some basic atomic operations in a classical algorithm with LLM-driven text-based ones while *retaining the high-level control flow of the classical algorithm*, thereby largely retaining the original theoretical guarantees.

This has advantages over another common approach to similar tasks that we call the "formalization-based approach". Leveraging the machine translation ability of LLMs, it converts an informal task representation into a formal, symbolic representation, solves it with a specialized combinatorial solver, and then converts the solution back to the informal, natural language representation for the user. Existing work based on this approach includes LLM-based generation of PDDL [McDermott, 2000] for classical planning [Xie, 2020, Guan et al., 2023, Li et al., 2024, Fine-Morris et al., 2024, Oswald et al., 2024], Prover9 input language for theorem proving [Olausson et al., 2023], or SMT solver input for optimization [Hao et al., 2024]. The issue with this approach is threefold: First, it requires grammar-following capability for a complex target formal language. Second, while the downstream solver requires a semantically and logically accurate input, the vagueness and informality in the natural language input often cause LLMs to generate inaccurate and inconsistent translations. Third, the assumptions made in the target formal language limits the scope of the tasks that can be solved by this approach. For example, a LLM-based planning agent that converts the task into PDDL cannot handle an informal task that contains probabilistic, temporal, or partially-observable nature. Verbalized algorithms, on the other hand, can handle them robustly via the general and informal common sense reasoning ability for atomic operations.

2 Verbalized Algorithms

Auto-regressive generative model is a probability distribution $p(\mathbf{x}) = p(\mathrm{x}_1, \dots, \mathrm{x}_N)$ where the random variable $\mathbf{x}$ represents a vector of integers, typically associated with a word or a sub-word token in natural language. Denoting its subsequence from index i to j as $\mathbf{x}_{i:j}$, Transformer-based language models (LMs) Vaswani et al. [2017] decompose it into each conditional distribution modeled by a large set of parameters θ: $p(\mathbf{x}) = p_\theta(\mathbf{x}_N|\mathbf{x}_{1:N-1})p(\mathbf{x}_{1:N-1}) = \ldots = \prod_{i=j}^{N-1} p_\theta(\mathbf{x}_i|\mathbf{x}_{1:i-1})p(\mathbf{x}_{1:j})$, where the prefix $\mathbf{x}_{1:j}$ is typically called a *prompt* and each conditional distribution $p_\theta(\mathbf{x}_i|\mathbf{x}_{1:i-1})$ performs a next-token prediction.

To exhibit a trait typically useful for user interaction (e.g. instruction following), LLMs go through a series of parameter-efficient fine-tuning (PEFT) steps after pretraining on a large corpus. For example, in Direct Preference Optimization [Rafailov et al., 2023, DPO], the input is a pair of model outputs in which human users have indicated their preference over another and the model is fine-tuned to follow that preference. For a prompt x and an answer pair y_1 and y_2, denote by $y_1 \succ y_2|x$ that y_1 is preferred over y_2. DPO learns a Bradley-Terry reward model [Bradley and Terry, 1952, BT-model] for this preference, i.e., $p(y_1 \succ y_2|x)$. The Plackett-Luce reward model [Plackett, 1975, Luce, 1959, PL-model] extends BT to a multinominal case that represents a ranking over choices.

While whether LLMs can reason or not is up for both philosophical and scientific debates, it is more widely accepted that they can solve some much simpler tasks with high fidelity, such as text-based classification and question-answering. This is analogous to say that general BT (and PL) reward models baked in the LLM are sufficient to perform such simple tasks. Assuming those capabilities and thus relying entirely on the accuracy of the reward models, we propose *verbalized algorithms* (VAs), an LLM-based reasoning strategy that limits the scope of the LLMs to simple tasks.

Definition 1 (Verbalized Algorithms)**.** *Let $T[\tau]$ be a computational task (e.g., sorting) over objects of type τ and let $A[\tau]$ be an algorithm that solves $T[\tau]$ with an n-arg boolean oracle $f[\tau] : \tau, \dots, \tau \to \{\top, \bot\}$. A* Verbalized Algorithm *instantiates $A[\tau]$ with $\tau =$ `str` by implementing $f[$`str`$]$ as a query to an LLM whose output is constrained to {"yes", "no"} which maps to $\{\top, \bot\}$.*

VAs keep the high-level control flow of $A[\tau]$ intact, while only replacing the oracle $f[\tau]$ with an LLM-based $f[$`str`$]$. This preserves the theoretical properties of the classical algorithm $A[\tau]$ if the LLM learned a perfect reward model that is equivalent to the ground truth. If the reward model is inaccurate, a non-classical robust algorithm that assumes noisy oracles can bound the error rate of the output using known theoretical results. VAs can also be combined with other specialized algorithms, such as concurrent algorithms. While our definition is focused on the BT reward model, it naturally extends to PL rewards / categorical oracles.

VAs have three categories: *Naive VAs*, which treat the LLM's answer as ground truth, *Robust VAs*, which treat it as a noisy oracle with a certain error rate and quantify the rate of error in the final output, and *Probabilistic VAs*, which use the probability/logits of LLM outputs to improve the accuracy, to quantify the uncertainty of the result, or to return a distribution of outputs.

3 Verbalized Sorting

Verbalized Sorting sorts a list of natural language strings X using LLM's question answering capability as an atomic comparison oracle. Each binary comparison for $x_1, x_2 \in X$ is replaced by a yes-no style LLM query such as "Is X better than Y? X: x_1 Y: x_2", where the next output token is restricted to "yes" and "no". The model output is mapped to $\top/\bot$ if the sampled output is "yes" / "no", respectively. This task is important in the RAG applications and also called the LLM-based ranking approach. We evaluated the accuracy of several non-VA and VA approaches. Method, dataset, and prompt details are available in the appendix Sec. A. All results are averages of 5 random seeds.

Constraint Decoding Baseline encodes the list X in a markdown format and asks the model to return a sorted list using structured-decoding [Willard and Louf, 2023] to constrain the output to be a list whose elements are guaranteed to be one of the input strings. Due to the lack of a computationally efficient way to constrain the list to (1) be free of duplicates and (2) represent the same set of strings as the input, we (1) remove the duplicates and (2) append missing elements to the end of the list. The approach corresponds to *listwise* ranking approach Ma et al. [2023] in the RAG literature.

Scoring-based approaches ask the model to assign a score between 1.0 and 5.0 (down to one decimal) to each sentence using regex-constrained decoding, then sort the sentences using the scores. It is a form of formalization-based approach (the list of scores represents the formal problem). We tested two variants: **I.i.d Scoring** scores $x_i \in X$ independently, and **Autoregressive Scoring** scores all strings in a single query, allowing the model to see the scores of other elements. The approach corresponds to *pointwise* ranking approach Sun et al. [2023], Sachan et al. [2022] in the RAG literature.

VA approaches correspond to *pairwise* ranking approaches Qin et al. [2024] in the RAG literature, but our contribution focuses on leveraging the theoretical properties of classical algorithms, which has been undermined in the literature. **VA Powersort** overrides Python's `__gt__` operator with an LLM call and applies the built-in `sorted` function, which uses Powersort [Munro and Wild, 2018] that runs $O(n \log n)$ comparisons worst-case, the best runtime complexity achievable by comparison sort. **VA Bitonic sorting network** uses the *Bitonic sorting network* [Batcher, 1968] from parallel sorting literature to improve the throughput. It performs $O(n(\log n)^2)$ comparisons, but runs in $O((\log n)^2)$ time due to parallelism. As an additional optimization, we use the optimal sorting network [Knuth, 1997, Bundala and Závodnỳ, 2014] when the input size is 16 during its recursion. **Robust VAs (Powersort/Bitonic)** make individual comparisons more robust by sampling the output $K = 3$ times as a batch LLM call and performing majority voting. For an odd K, if the oracle is correct with probability p, the voting result is correct with at least probability $1 - e^{-K(2p-1)^2/2}$ [Emamjomeh-Zadeh and Kempe, 2018] due to Hoeffding bound [Hoeffding, 1963]. Finally, **Probabilistic VAs (Powersort/Bitonic)** instead rely on comparing the probabilities / logits over the yes/no answers to perform each comparison query. While the naive probabilistic VA is equivalent to the naive VA with greedy sampling, we additionally have **Symmetrized VA** to address two biases in LLM: Positional bias and psycophancy (the tendency to reply "yes") . Let p_{12} and p_{21} be next-token probabilities for answering "yes" with the original prompt and a prompt whose x_1 and x_2 are swapped. Due to the bias, $p_{12} \neq 1 - p_{21}$. Symmetrized VA tests $\frac{p_{12}+(1-p_{21})}{2} > \frac{(1-p_{12})+p_{21}}{2}$ as a comparison oracle.

Evaluation We evaluated the approaches above on Amazon reviews benchmark [Hou et al., 2024]. We selected 30 products from the "Video Games" category, which, for each integer rating between 1 and 5, have at least 5 non-empty reviews that are voted as "helpful" by other customers. The task is to sort a shuffled list of 25 selected reviews (5 for each rating) in increasing order of positive sentiment. To measure the accuracy, we compute the Kendall-Tau score [Kendall, 1938] between the output list Y and the ground-truth output list Y^*, which is sorted by the true rating, averaged over the dataset. We used Qwen3-1.7B and Qwen3-32B-AWQ models [Team, 2025] with thinking enabled.

Table 1: Average number of sorting invariance violations by the constraint decoding baseline. "Duplicates" indicates the number of duplicates found in the output, and "missing" indicates the number of input elements that are missing in the output, for 25 input strings.

Models (Qwen3)	duplicates	missing
1.7B	1.39	9.63
32B-AWQ	0.49	2.75

First, Table 1 shows how often the constraint-decoding baseline violates the sorting invariance, i.e., how many elements are duplicated or missing in the output on average. While these violations become rarer as the model gets larger, they do not disappear. This highlights the lack of theoretical guarantees in LLM-based reasoning, which does not occur in VAs and formalization-based approaches.

Next, Table 2 shows the average Kendall-Tau scores. Constraint-decoding baseline performed the worst overall, and while the scoring approaches can work with the large 32B model, their performance dwindles when the model is smaller. Robust VA with 32B achieved the best results. *The strength of VA is most pronounced with the small 1.7B model*, where other approaches failed. Notably, VAs with a 1.7B model outperformed the baseline with a 32B model, clearly showing the advantage of using the sorting algorithms developed in the literature.

Finally, Table 3 compares the runtime of each approach. VA Bitonic Sorting Network improves over the VA Powersort, and both VAs can be faster than the constrained-decoding baseline, even with robustification (x3 parallel queries) or symmetrization (x2 parallel queries). This shows that VAs can directly benefit from the theoretical results in the parallel sorting literature. Note that the parallel speedup in bitonic sort does not reach the theoretical limit (x11.4; 171 comparators and 15 steps for 25 inputs) because the maximum batch size is limited by the GPU; multi-GPU experiments with data-parallelism would improve this.

The Effect of Prefix Caching Each query in VA sort requires a single next-token prediction. In Transformer architecture, it requires $O(t^2)$ computations for t input tokens. However, State-of-the-Art inference engines such as vllm Kwon et al. [2023] and SGLang Zheng et al. [2024] cache the *prefix kv-cache* over multiple queries. This is particularly helpful in reducing the inference cost of VA-style queries majority of which share the prefix.

A comparison query in VA sorting consists of a prompt, x_1, and x_2. Assume all strings being compared has a fixed length $L = |x_1| = |x_2|$, and that the length of the fixed prefix is L_p. In the exhaustive pairwise comparison among n strings ($O(n^2)$ comparisons, equivalent to insertion sort etc.), a naive inference engine requires $O((L_p + 2L)^2)$ runtime per each yes/no query, thus $O(n^2(L_p + 2L)^2)$ runtime in total, without parallelism.

Assume that an input consists of $N + M$ tokens (prefix and suffix), and the kv-cache of N prefix tokens is already available. Each $N + t$-th new input token attends to $N + (t - 1)$ existing tokens, which makes $N + M + 1$-th token require $\sum_{t=1}^{M}(N + (t - 1)) = O(NM + M^2)$ runtime.

With this in mind, assuming that the fixed prompt already has a kv-cache, the runtime of each comparison query reduces to $O(2L_pL+4L^2)$, totalling $O(L_p^2+n^2(2L_pL+4L^2))$ runtime. Moreover, since x_1 is repeated n times over n^2 comparisons, it further reduces to $O(L_p^2 + n(L_pL + L^2) + n^2((L_p + L)L + L^2)) = O(L_p^2 + n(L_pL + L^2) + n^2(L_pL + 2L^2))$. This is significant because the prompt tends to be much longer than each string.

Table 4 demonstrates the empirical speedup due to kv-cache in two State-of-the-Art inference engines. RadixAttention in SGLang is particularly helpful for VAs whose queries share large prefixes.

4 Verbalized Clustering

Verbalized Clustering attempts to cluster a set of strings S into a family of subsets $\mathcal{S}$ based on their similarity. The problem with text-based clustering is that the distance function for natural language strings is difficult to define. All approaches to this task are embedding methods. Method, dataset, and prompt details are available in the appendix Sec. B. All results are averages of 5 random seeds.

Table 2: Average Kendall-Tau scores over 5 seeds. Best scores in **bold**. The score of 1, 0, and -1 indicates perfect rank correlations, no correlations, and perfect anti-correlations, respectively.

Models		Scoring		Naive VA		Robust ($K = 3$)		Symmetrized	
(Qwen3)	Baseline	i.i.d.	AR	Bitonic	Powersort	Bitonic	Powersort	Bitonic	Powersort
1.7B	0.00	0.18	0.05	0.30	0.33	0.33	0.44	0.33	**0.48**
32B-AWQ	0.30	0.57	0.39	0.39	0.56	0.39	0.58	0.41	**0.60**

Table 3: Average total elapsed time in seconds for processing 100 sorting tasks (each with 25 strings). Bitonic sorting network also shows the relative speedup over the sequential Powersort.

Models		Scoring		Naive VA		Robust ($K = 3$)		
(Qwen3)	Baseline	i.i.d.	AR	Power.	Bitonic	Power	Bitonic	Bitonic+16
1.7B	9279	636	194	5196	2476 (x2.1)	7279	5304 (x1.4)	4728 (x1.5)
32B-AWQ	6798	1406	661	18820	8380 (x2.3)	25302	20461 (x1.2)	17169 (x1.5)

Jaccard Baseline creates a distance matrix using Jaccard distance of token sets of strings, then applies `sklearn`'s MDS [Kruskal, 1964] to create an embedding. **Sentence Transformer (ST) Baseline** uses Qwen3-8B-Embedding directly, which was selected as the top performer in the MTEB leaderboard [Muennighoff et al., 2022]. **VA t-STE:** This is a VA based on Student-t Stochastic Triplet Embedding [Van Der Maaten and Weinberger, 2012, t-STE], which makes *triplet comparison queries* to LLM. Triplet comparison takes three elements x, y and z in a metric space $\|\cdot\|$ and query if $\|x-y\| < \|x-z\|$. We ask the LLM a question such as "Is X closer to Y than is to Z? X: x, Y: y, Z: z". For each $x \in S$, we randomly sample k queries from S^3 and their results from an LLM, then generate D-dim embeddings by minimizing the number of violations to the results. We show the results with hyperparameters $k \in \{100, 200\}, D = 8$.

We evaluated the methods across four datasets. **Medarxiv** [Muennighoff et al., 2022] contains the titles and the categories of medical papers on MedArxiv. The category forms a cluster. **IT Field Defects** consists of 78 technical IT field defect reports submitted to our internal defect tracking system (unpublished), where some reports are manually marked as duplicates of another, forming a cluster. **xkcdcolors-h/l** are two datasets of 100 colors sampled from [Munroe, 2010]. Each entry has a non-standard color name (e.g., `baby blue`, `vomit`) and a corresponding RGB value. The difference between **h** and **l** variants is that we defined 6 ground truth clusters with `sklearn.AgglomerativeClustering` using a lexicographic distance that prioritizes *hue* and *lightness* in the HSL color model, respectively. Assuming that colors tend to be more strongly associated with hues, xkcdcolors-l represents an out-of-distribution (OOD) task.

We evaluate the embeddings with a score defined in Eq. 1, which is the average rank of the members of the cluster each string belongs to, when sorted by the Euclidean distance. The score is normalized to $[0, 1]$, where 1 indicates that cluster members are embedded closer. Given the input S and a string $s \in S$, let $C(s)$ be the ground-truth cluster such that $s \in C(s)$. Let $i = \text{RANK}(s, s') \in \{0, 1, \dots |S| - 1\}$ be an integer such that, when sorted by the pairwise Euclidean distance $d(e(s), e(s''))$ between s and a string s'' across $s'' \in S$ using embedding e, then $d(e(s), e(s'))$ ranks i-th. The score is defined as:

$$\textit{score} = \frac{1}{|S|} \sum_{s \in S} \frac{1}{|C(s)| - 1} \sum_{s' \in C(s) \setminus \{s\}} 1 - \frac{\text{RANK}(s, s')}{|S|} \in [0, 1] \tag{1}$$

Table 5 shows the results. Jaccard fared surprisingly well in IT Defects, where the input often contains long output logs whose tokens carry significant information, but it underperformed in tasks

Table 4: Average total elapsed time of naive Powersort in seconds, comparing VLLM without kv-cache sharing, with kv-cache sharing via PagedAttention, and SGLang with RadixAttention.

Models	VLLM (PagedAttention)		SGLang
(Qwen3)	disabled	enabled	(RadixAttention)
1.7B	5196	5403 (x.96)	3812 (x1.4)
32B-AWQ	18820	17421 (x1.1)	14424 (x1.3)

Table 5: Clustering scores across methods and benchmarks. The top 2 results are highlighted in bold.

method	model	$D =$	$k =$	Medarxiv	IT Defects	xkcdcolors-h	xkcdcolors-l
Jaccard		8		0.556	0.767	0.601	0.472
ST	Qwen3-Emb...d-8B	4096		0.827	**0.936**	0.718	0.531
triplet	Qwen3-4B	8	100	0.831	0.831	0.823	0.704
triplet	Qwen3-4B	8	200	**0.858**	0.895	0.823	0.691
triplet	Qwen3-32B-AWQ	8	100	0.842	0.908	**0.851**	**0.748**
triplet	Qwen3-32B-AWQ	8	200	**0.858**	**0.939**	**0.850**	**0.742**

that require semantic comprehension. Sentence Transformer achieves strong performance in the standard clustering tasks such as Medarxiv and IT Defects. However, it fell short in xkcdcolors-h/l that requires information about the distance criteria. The score 0.53 in the xkcdcolors-l is close to 0.5, i.e., its distance ranking is only marginally better than a dice roll. VA t-STE outperformed traditional baselines with $k = 200$. We highlight its strong generalization in OOD tasks where other approaches failed, due to its strong algorithmic bias combined with flexible LLM queries.

5 Related Work

Despite significant recent findings in reasoning models, which combine RL-based self-training with "think tags" [Guo et al., 2025], they rarely explicitly perform systematic backtracking and search.

Whether such systems can reason or not has been hotly debated in the context of LLM-based planning [Valmeekam et al., 2023]. Previous work on solving planning tasks using LLM [Liu et al., 2023, Hirsch et al., 2024] demonstrates varying degrees of success and limitations (e.g., failure in `mystery` due to sensitivity to the vocabulary). Formalization approach to planning is also called a *parse-and-solve* model [Collins et al., 2022] as opposed to a *LLM-as-Planner* model that naively generates a plan with an LLM. Thought-of-Search [Katz et al., 2024] simulates a systematic search algorithm, which could be seen as an instance of VAs. To our knowledge, there is no work that simulates traditional graph search algorithms (e.g., A* [Hart et al., 1968], GBFS [Bonet and Geffner, 2001]) in a VA manner, i.e., by replacing all of its atomic operations (e.g., successor generation, heap-based priority queues, hash computation for duplicate detection) with LLM queries, although some work replace some atomic operations, especially successor generations [Katz et al., 2024].

A number of recent papers in the *Algorithm with Predictions* [Mitzenmacher and Vassilvitskii, 2022, ALPS] community achieved a significant success in augmenting algorithms by data-driven oracles. The particular strength of this line of work is the rigorous theoretical analysis of the error rate. VAs is a form of ALPS specialized in LLM-based oracles, thus combining ideas from ALPS community with VAs is a promising direction for future work.

Combining data-driven heuristics with existing algorithms in the classical AI tasks is also a common practice in the recent literature (e.g., classical planning Núñez-Molina et al. [2024], boolean SAT Duan et al. [2020], logistics tasks Xin et al. [2021]).

6 Conclusion, Discussion and Future Work

We presented *Verbalized Algorithms* (VAs), a disciplined paradigm for performing algorithmic reasoning with LLMs. It benefits from a strong prior in the golden classics of discrete algorithms published half a century ago, significantly improving the reasoning accuracy even with a small model. It also leverages their established theoretical results to improve the runtime of such reasoning.

While this paper explored only a limited number of applications due to space, we are already pursuing several exciting verbalizations, including submodular maximization and pareto-front computation for diverse / non-dominated document retrieval, DPLL for combinatorial reasoning, and median/top-k selection, etc. The potential of this paradigm is large, and we invite various ideas for unique verbalizations and new applications. For example, can Floyd's tortoise-and-hare cycle detection algorithm for linked lists be verbalized to perform a tautology detection? Garbage collection algorithms to detect unimportant sentences in a paper draft?

References

K. E. Batcher. Sorting Networks and Their Applications. In *Spring Joint Computer Conference*. ACM, 1968.

B. Bonet and H. Geffner. Planning as Heuristic Search. *Artificial Intelligence*, 129(1):5–33, 2001.

R. A. Bradley and M. E. Terry. Rank Analysis of Incomplete Block Designs: I. The Method of Paired Comparisons. *Biometrika*, 39(3/4):324–345, 1952.

D. Bundala and J. Závodnỳ. Optimal Sorting Networks. In *International Conference on Language and Automata Theory and Applications*, pages 236–247. Springer, 2014.

K. M. Collins, C. Wong, J. Feng, M. Wei, and J. B. Tenenbaum. Structured, flexible, and robust: benchmarking and improving large language models towards more human-like behavior in out-of-distribution reasoning tasks. *arXiv preprint arXiv:2205.05718*, 2022.

H. Duan, S. Nejati, G. Trimponias, P. Poupart, and V. Ganesh. Online Bayesian Moment Matching Based Sat Solver Heuristics. In *Proc. of the International Conference on Machine Learning (ICML)*, pages 2710–2719. PMLR, 2020.

E. Emamjomeh-Zadeh and D. Kempe. Adaptive Hierarchical Clustering using Ordinal Queries. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 415–429. SIAM, 2018.

M. Fine-Morris, V. Hsiao, L. N. Smith, L. M. Hiatt, and M. Roberts. Leveraging LLMs for Generating Document-Informed Hierarchical Planning Models: A Proposal. In *AAAI 2025 Workshop LM4Plan*, 2024.

L. Guan, K. Valmeekam, S. Sreedharan, and S. Kambhampati. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, 36:79081–79094, 2023.

D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al. Deepseek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arXiv preprint arXiv:2501.12948*, 2025.

Y. Hao, Y. Zhang, and C. Fan. Planning Anything with Rigor: General-Purpose Zero-Shot Planning with LLM-based Formalized Programming. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2024.

P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

E. Hirsch, G. Uziel, and A. Anaby-Tavor. What's the Plan? Evaluating and Developing Planning-Aware Techniques for Language Models. *arXiv preprint arXiv:2402.11489*, 2024.

W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Y. Hou, J. Li, Z. He, A. Yan, X. Chen, and J. McAuley. Bridging Language and Items for Retrieval and Recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

M. Katz, H. Kokel, K. Srinivas, and S. Sohrabi. Thought of Search: Planning with Language Models Through The Lens of Efficiency. In *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, volume 37, page 138491–138568. Curran Associates, Inc., 2024.

M. G. Kendall. A New Measure of Rank Correlation. *Biometrika*, 30(1-2):81–93, 1938.

D. E. Knuth. *The Art of Computer Programming. Vol. III: Sorting and Searching*. 1997.

J. B. Kruskal. Nonmetric Multidimensional Scaling: A Numerical Method. *Psychometrika*, 29(2):115–129, 1964.

W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica. Efficient Memory Management for Large Language Model Serving with Pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

R. Li, L. Cui, S. Lin, and P. Haslum. Naruto: Automatically acquiring planning models from narrative texts. In *Proc. of AAAI Conference on Artificial Intelligence*, volume 38, pages 20194–20202, 2024.

B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency. *arXiv preprint arXiv:2304.11477*, 2023.

R. D. Luce. *Individual Choice Behavior: A Theoretical Analysis*, volume 4. Wiley New York, 1959.

X. Ma, X. Zhang, R. Pradeep, and J. Lin. Zero-Shot Listwise Document Reranking with a Large Language Model. *arXiv preprint arXiv:2305.02156*, 2023.

D. V. McDermott. The 1998 AI Planning Systems Competition. *AI Magazine*, 21(2):35–55, 2000.

M. Mitzenmacher and S. Vassilvitskii. Algorithms with Predictions. *Communications of the ACM*, 65(7): 33–35, 2022.

N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. MTEB: Massive Text Embedding Benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

J. I. Munro and S. Wild. Nearly-Optimal Mergesorts: Fast, Practical Sorting Methods That Optimally Adapt to Existing Runs. In *Annual European Symposium on Algorithms*, 2018.

R. Munroe. Color survey results, 2010.

C. Núñez-Molina, M. Asai, P. Mesejo, and J. Fernández-Olivares. On Using Admissible Bounds for Learning Forward Search Heuristics. In *Proc. of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 6761–6769, 2024.

T. X. Olausson, A. Gu, B. Lipkin, C. E. Zhang, A. Solar-Lezama, J. B. Tenenbaum, and R. P. Levy. LINC: A Neurosymbolic Approach for Logical Reasoning by Combining Language Models with First-Order Logic Provers. In *Proc. of Empirical Methods in Natural Language Processing*, 2023.

J. Oswald, K. Srinivas, H. Kokel, J. Lee, M. Katz, and S. Sohrabi. Large Language Models as Planning Domain Generators. *Proceedings of the International Conference on Automated Planning and Scheduling*, 34(1):423–431, May 2024. doi: 10.1609/icaps.v34i1.31502.

R. L. Plackett. The Analysis of Permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 1975.

Z. Qin, R. Jagerman, K. Hui, H. Zhuang, J. Wu, L. Yan, J. Shen, T. Liu, J. Liu, D. Metzler, et al. Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1504–1518, 2024.

R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, 36:53728–53741, 2023.

D. Sachan, M. Lewis, M. Joshi, A. Aghajanyan, W.-t. Yih, J. Pineau, and L. Zettlemoyer. Improving Passage Retrieval with Zero-Shot Question Generation. In *Proc. of Empirical Methods in Natural Language Processing*, pages 3781–3797, 2022.

W. Sun, L. Yan, X. Ma, S. Wang, P. Ren, Z. Chen, D. Yin, and Z. Ren. Is Chatgpt Good at Search? Investigating Large Language Models as Re-Ranking Agents. In *Proc. of Empirical Methods in Natural Language Processing*, pages 14918–14937, 2023.

Q. Team. Qwen3 Technical Report, 2025. URL `https://arxiv.org/abs/2505.09388`.

K. Valmeekam, M. Marquez, S. Sreedharan, and S. Kambhampati. On the Planning Abilities of Large Language Models - A Critical Investigation. *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, 36:75993–76005, 2023.

L. Van Der Maaten and K. Weinberger. Stochastic Triplet Embedding. In *2012 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2012.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is All You Need. In *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, pages 5998–6008, 2017.

B. T. Willard and R. Louf. Efficient Guided Generation for Large Language Models. *arXiv preprint arXiv:2307.09702*, 2023.

Y. Xie. Translating natural language to planning goals with large-language models. *The International Journal of Robotics Research*, 2019:1, 2020.

L. Xin, W. Song, Z. Cao, and J. Zhang. Neurolkh: Combining Deep Learning Model with Lin-Kernighan-Helsgaun Heuristic for Solving the Traveling Salesman Problem. *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, 34:7472–7483, 2021.

L. Zheng, L. Yin, Z. Xie, C. L. Sun, J. Huang, C. H. Yu, S. Cao, C. Kozyrakis, I. Stoica, J. E. Gonzalez, et al. SGLANG: Efficient Execution of Structured Language Model Programs. *Proc. of the Advances in Neural Information Processing Systems (Neurips)*, 37:62557–62583, 2024.

Appendix

A Sorting: Experimental Details

All methods uses a common shared criteria for comparison. In the Amazon dataset used in the study, the criteria prompt is:

```
criteria = ‘‘You will be given two product reviews, X and Y, written
by various customers about the same product. Is X more favorable to
the product than Y?’’
```

We used the following query prompts:

Constrained-decoding baseline: `f"Sort the following list in the increasing order of quality that is compared in the following comparison criteria: '{criteria}'. That is, for a pair X and Y, if the answer to the criteria is 'yes', X must appear after Y in the output list. "`

I.i.d Scoring: `f"I want to sort several strings. Given two strings X and Y, my comparison criteria is as follows: '{criteria}'. Assign a score to X below so that for any pair of strings X and Y, if the answer to the comparison criteria above is yes, the score of X is greater than the score of Y. Each score must be between 1 and 5 with one decimal place, where the score 5 means that X will most certainly satisfy the comparison against other strings, and the score 1 means the opposite. For example, if we compare the height of mountains, score 5 will be assigned to very tall mountains, and if we compare the quality of restaurants, score 1 implies that the food or the service is very bad. \nX: "` followed by a string.

Autoregressive Scoring: `f"I want to sort several strings listed below. Given two strings X and Y, my comparison criteria is as follows: '{criteria}'. Assign a score to each string so that for any pair of strings X and Y, if the answer to the comparison criteria above is yes, the score of X is greater than the score of Y. Return the scores as a space-separated list. Each score must be between 1 and 5 with one decimal place, where the score 5 means that X will most certainly satisfy the comparison against other strings, and the score 1 means the opposite. For example, if we compare the height of mountains, score 5 will be assigned to very tall mountains, and if we compare the quality of restaurants, score 1 implies that the food or the service is very bad. \n strings:"` followed by a markdown-formatted list of strings.

Verbalized Sorting: `f"{criteria} Answer with a single word, yes or no. X: '{self.string}' Y: '{other.string}'"`

B Clustering: Experimental Details

Triplet comparison query uses the following shared prompt, where `criteria` is a dataset-specific prompt.

```
f"{criteria} \nX: t.x\nY: t.y\nZ: t.z\n Answer either 'X' or 'Y'."
```

The dataset-specific prompt is as follows:

Medarxiv dataset: `criteria = f"You will be given three paper titles X, Y, and Z. Does the paper Z discuss a topic that is closer to the topic in X than to the topic in Y?"`

IT Defect dataset: `criteria = f"You will be given three defect reports X, Y, and Z, sent by the customers of our GPFS filesystem storage system. They may contain stack traces of various programs (python, kernel drivers, ...). Similar repetitive lines are omitted as [...]. Considering the cause of the defects (i.e. ignoring timestamp/IP/MAC addresses), are X and Z more similar than Y and Z are? "`

xkcdcolors-h dataset: `criteria = f"Which of the following two colors, X and Y, is more similar to the color Z? Focus on the hue in the HSL model first, then other aspects such as lightness and saturation. For example, Red and Pink are more similar than Red and Blue are. If you are not familiar with the given color name, e.g., a made-up color name like 'puke brown', try to imagine what the color would look like, e.g., puke would generally have a lighter shade. "`

xkcdcolors-l dataset: `criteria = f"Which of the following two colors, X and Y, is more similar to the color Z? Focus on the lightness in the HSL model first, then other aspects such as hue and saturation. For example, Red and Blue (both lightness = 50If you are not familiar with the given color name, e.g., a made-up color name like 'puke brown', try to imagine what the color would look like, e.g., puke would generally have a lighter shade. "`

B.1 IT Defect Reports Dataset

Let D denote the set of all defects. A subset $D' \subseteq D$ was manually labeled as duplicates of several other defects in D. We selected 78 data points so that the final dataset contains 25 such clusters, each of size at least 2. The following is an example of a redacted/simplified defect record that appear in the dataset.

```
{
  "id": 347852,
  "description":
  "System login:
    EMS - X.XX.XXX.XXX root/cluster
    S6K w/ Falcon attached - lothal-qa6-[1 && 2] Code levels:
    S6k - 6230_B19
    FCM4 - 4_3_3
    Falcon - 5342 --> 5343

   Snap location:
    glogin:/u/DUMPS/R.347852/

   What I was doing:
    I was upgrading Falcon enclosure from 5342 to 5343.

   What I expected:
    'XXXXfirmware' will start and the Falcon will go through the upgrade process.

   What happened:
    'mmchrimware' completed but there are errors printed.
    The Falcon does appear to have the updated firmware.
    However, there are issues when trying to get a print
    from the Left module when running 'essenclosures'.

   Analysis:
    sg91:13:/dev/sg91::naa.500093d006585000/10910::[1.0.91.0]:/dev/sg91:XXX:5149-091:5342
    [10:48:31.258608008] XXXXXXFirmwareVer2 checking if enclosure needs to be updated
    [10:48:31.260567945] XXXXXXFirmwareVer2 firmware file found
    [10:48:31.262950549] XXXXXXFirmwareVer2 -> ...
    [10:48:31.350737778] XXXXXXFirmwareVer2 ...
    [10:48:31.362309567] XXXXXXFirmwareVer2 lothal-qa6-1-hs: [E] Error 1 updating firmware ...
    sg_ses failed: Transport error
    ...",
  "duplicate_of": [],
  ...
}
```