Decomposed Prompting: A MODULAR APPROACH FOR SOLVING COMPLEX TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Few-shot prompting is a surprisingly powerful way to use Large Language Models (LLMs) to solve various tasks. However, this approach struggles as the task complexity increases or when the individual reasoning steps of the task themselves are hard to learn, especially when embedded in more complex tasks. To address this, we propose Decomposed Prompting, a new approach to solve complex tasks by decomposing them (via prompting) into simpler sub-tasks that can be delegated to a shared library of prompting-based LLMs dedicated to these sub-tasks. This modular structure allows each prompt to be optimized for its specific sub-task, further decomposed if necessary, and even easily replaced with more effective prompts, trained models, or symbolic functions if desired.

We show that the flexibility and modularity of Decomposed Prompting allows it to outperform prior work on few-shot prompting using GPT-3. On symbolic reasoning tasks, we can further decompose sub-tasks that are hard for LLMs into even simpler solvable sub-tasks. When the complexity comes from the input length, we can recursively decompose the task into the same task but with smaller inputs. We also evaluate our approach on textual multi-step reasoning tasks: on long-context multi-hop QA, we can more effectively teach the sub-tasks via our separate sub-tasks prompts; and on open-domain multi-hop QA, we can easily incorporate a symbolic information retrieval module within our decomposition framework, leading to improved performance on both tasks.[1]

1 INTRODUCTION

Large Language Models (LLMs) such as GPT-3 (Brown et al., 2020) have been shown to solve various tasks given only a few examples as prompts, also referred to as in-context learning. These models can even perform more complex reasoning tasks when shown the sequence of simple reasoning steps needed to perform the complex task as a prompt Wei et al. (2022); Nye et al. (2021). In essence, the sequence of reasoning steps, such as in Chains-of-Thought (CoT) prompting (Wei et al., 2022), demonstrates how to decompose the complex task as well as how each reasoning step should be performed. However, as tasks become more complex, few demonstrations of the complex task aren't sufficient for current models to learn to perform all necessary reasoning steps. E.g., few-shot demonstrations of concatenating the k^{th} letter of words in a string is insufficient for GPT-3 to learn to extract the k^{th} letter, or learn to answer hard single-hop questions when only provided a few demonstrations of multi-hop questions. Additionally, it is unclear whether tasks such as document retrieval and integration, for knowledge-intensive tasks, can even be done by few-shot prompts.

To address these limitations, we propose **Decomposed Prompting** (DECOMP), a new approach to solve complex tasks by instead decomposing them into simpler sub-tasks and delegating these to sub-task specific LLMs, with both the decomposer and the sub-task LLMs (henceforth, *sub-task handlers*) having their own few-shot prompts. Fig 1 illustrates our approach. The decomposer prompt only describes a sequence of sub-tasks (A, B, and C) needed to solve the complex tasks, indicated with the dashed lines. Each sub-task is then delegated to the corresponding sub-task handler shown on the right.

[1] Datasets, Code and Prompts available at ANONYMOUS

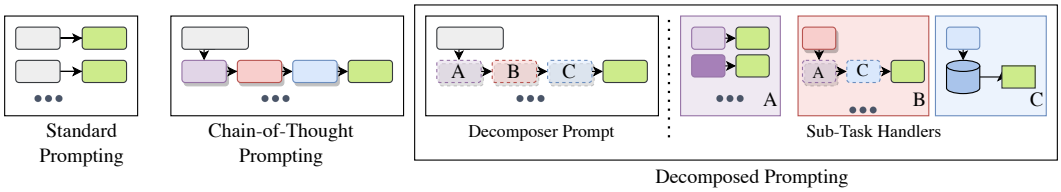


Figure 1: While standard approaches only provide labeled examples (shown as a grey input box with green label box), Chain-of-Thought prompting also describes the reasoning steps to arrive at the answer for every example in the prompt. Decomposed Prompting, on the other hand, uses the decomposer prompt to only describe the procedure to solve the complex tasks using certain sub-tasks. Each sub-task, indicated here with A, B and C is handled by sub-task specific handlers which can vary from a standard prompt (sub-task A), a further decomposed prompt (sub-task B) or a symbolic function such as retrieval (sub-task C)

Using a software engineering analogy, the decomposer defines the top-level *program* for the complex task using interfaces to simpler, sub-task functions. The sub-task handlers serve as modular, debuggable, and upgradable *implementations* of these simpler functions, akin to a software library. If a particular sub-task handler, say the one for identifying the k^{th} letter or retrieving a document, is not performing well enough, we can debug this handler in isolation, explore alternative prompts or implementations, and seamlessly plug the improved module back into the overall system, as a systematic way to try to improve performance on the complex end-task.

This approach has several advantages over prior work (as also shown in the figure). The sub-task handlers can be shown a broader and richer set of examples (of the simpler task) than the specific ones needed for the complex task prompt (task A). If a sub-task is too complex, it can be further decomposed into simpler sub-tasks (task B). Similar to software libraries, these sub-task handlers can be shared across multiple tasks; e.g., here tasks A and C are reused in the model for task B. As noted above, a sub-task handler can be easily swapped with an improved implementation without any change to the rest of the system. Finally, few-shot prompt based LLMs can be even replaced with a symbolic system for tasks more suited for non-neural methods; e.g., task C uses a symbolic retrieval system such as Elasticsearch that can handle very large-scale corpora.

To illustrate these advantages of DECOMP, we empirically evaluate it against prior work on four challenging datasets using GPT3 models: 1) On a task of concatenating the k^{th} letter, we show that our approach of factoring out each sub-task allows us to more effectively teach the sub-problem of extracting the k^{th} letter(specifically, by decomposing it into even easier sub-tasks). 2) On a task of reversing a list, we show that DECOMP allows us to extend the capabilities of a weaker model and build a scale-invariant system by recursively decomposing the task into reversal of smaller and smaller lists. 3) On a task of long-context QA (Khot et al., 2022), our approach allows each sub-task handler to accommodate more examples than feasible with CoT prompting leading to better QA performance. 4) On multi-hop open-domain QA dataset Yang et al. (2018), we can incorporate a symbolic retrieval (ElasticSearch) API as the handler for the retrieval sub-task leading to better results than CoT systems.

2 RELATED WORK

Few-shot Prompts for Multi-Step Reasoning Large-scale Language models (LLMs) have been shown to learn various NLP tasks given just few examples as prompts (Brown et al., 2020). Recently, they have also been successfully applied to various multi-step reasoning tasks by providing the intermediate reasoning steps, i.e. Chain-of-Thought (Wei et al., 2022; Chowdhery et al., 2022), needed to arrive at the answer. Building of the successes of Chain-of-thought (CoT) style prompting on LLMs, many subsequent studies have focused on improvements of this basic model. This includes, for example, diverse decoding and ensembling to improve self-consistency (Wang et al., 2022c), more complex and contrasting rationales (Marasović et al., 2021; Wang et al., 2022b), generating sub-questions and answering them sequentially in least-to-most complexity order (Zhou et al., 2022), integration with symbolic modules (Jung et al., 2022), selection inference (Creswell et al., 2022), coupling in-context learning with sparse fine-tuning (Zelikman et al., 2022), and even presented many of them under a single probabilistic programming framework (Dohan et al., 2022).

We view these prior works as specialized systems with a pre-defined decomposition structure. For example, the closest work to our approach is the idea of least-to-most prompting (Zhou et al., 2022) where one prompt is used to generate the sub-questions needed to answer a complex question and a second prompt sequentially answers these sub-questions. In contrast, our approach allows for diverse decomposition structures and can use different prompts to answer each sub-question, even including symbolic systems. Moreover, rather than generating the entire set of sub-questions (i.e., the decomposition) at once, we can iteratively generate new questions based on the answers to previous steps (e.g., our prompt for HotpotQA in Sec. 4.4 is one such use case.)

Modular Approaches for Multi-Step Reasoning Our work follows a long literature in NLP on neural modular modeling architectures (Andreas et al., 2016; Talmor & Berant, 2018; Min et al., 2019; Jiang & Bansal, 2019; Gupta et al., 2020; Perez et al., 2020; Khot et al., 2021; Levine et al., 2022) for question-answering and other tasks. We take particular inspiration from the *Text Modular Networks* approach of Khot et al. (2021), whereby problem decomposition consists of a learned *next question* generator trained to generate questions in the language of a collection of textual and symbolic agents. Best-first search strategy was used to explore the space of possible decompositions during inference. In contrast to this work, which largely centered around supervised training of the next-question generator *given existing agents*, we leverage the power and recent successes of few-shot LLMs to build both the decomposer and the sub-task agents that best fit the ideal decomposition. This has the advantage of obviating the need for specialized supervised training data that may not always be available for all sub-tasks – a key bottleneck of this prior work.

Neural Programming Our work also relates to work on neural programming architectures (Reed & De Freitas, 2016; Neelakantan et al., 2016; Cai et al., 2017), which aims at building neural models that can represent and execute compositional programs, often focusing on algorithmic tasks of the type we investigate here. Specifically, our top-level decomposer prompts with their corresponding *sub-task handlers* (see Figure 1), which get constructed incrementally and interpreted by the underlying LLM, can be interpreted as structured programs where the the underlying program construction and interpretation process is performed using LLMs.

3 DECOMPOSED PROMPTING

As with conventional *few-shot* prompting, the goal is to teach an LLM to find an answer A to a query Q using a small set of *in-context* examples $D = \{E_1, ..., E_{|D|}\}$. The answer A is obtained from the underlying distribution $p(A \mid Q, D, \theta)$ (Dohan et al., 2022). In the most basic few-shot setup, examples take the form $E_j = (Q_j, A_j)$. In the case of CoT-style prompting, the goal is to obtain answers by first generating a sequence or chain of intermediate reasoning steps or "thoughts" T, and then deriving the final answer based on T. To teach this ability, one uses more sophisticated in-context examples that take the form $E_j = (Q_j, (T_{j,1}, \ldots, T_{j,k}), A_j)$.

In DECOMP, the core is a *decomposer* LLM that tries to solve a complex task by generating a **prompting program** P for it. Each step of P directs a simpler sub-query to a function in an auxiliary set of **sub-task functions** $\mathcal{F}$ available to the system. Given a query Q whose answer is A, the program P is a sequence of the form $\big((f_1, Q_1, A_1), ..., (f_k, Q_k, A_k)\big)$ where A_k is the final answer predicted by the program and Q_i is a sub-query directed to the sub-task function $f_i \in \mathcal{F}$. The prompting program P is executed by a high-level imperative **controller**, which passes the inputs and outputs between the decomposer and sub-task handler until a stopping condition is met in P and a final output is obtained.

To teach the decomposer LLM in a few-shot prompting manner, we use in-context examples that take the form $E_j = \big((Q_j, \big(f_{j,1}, Q_{j,1}, A_{j,1}), ..., (f_{j,k_j}, Q_{j,k_j}, A_{j,k_j})\big)\big)$ where $A_{j,k_j} = A_j$ is the final answer for Q_j and $(Q_{j,1}, \ldots, Q_{j,k_j})$ is a decomposition of Q_j. Each sub-task function f, in turn, is operationalized via a sub-task handler as an in-context prompting LLM (e.g., a separate CoT-style prompt or a additional prompting program dedicated to that sub-task), or any other symbolic or learned function (e.g., a calculator or specialized supervised trained model).

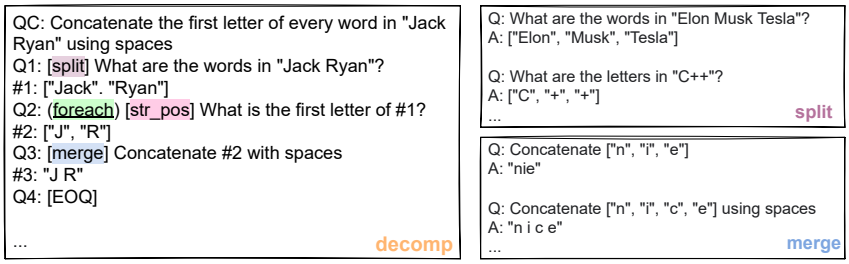


Figure 2: Prompts for the decomposer and the `split` and `merge` sub-tasks used by the decomposer. The decomposer specifies the sequence of questions and corresponding sub-tasks (within square braces). The sub-task prompts can be written independent of the complex task examples and can even capture generalizations, e.g., letters in word (`split`) and no delimiter (`merge`).

3.1 DECOMPOSED PROMPTS

To illustrate this with an example, consider a multi-step task such as "Concatenate the first letter of every word in *str* using a space". We can solve this task by decomposing it into a sequence of three simple sub-tasks: 1) Collect the list of words in the *str*; 2) For each word, extract the third letter; 3) Concatenate the extracted letters using space as the separator. Figure 2 shows an example decomposition prompt for this task. Much like a conventional structured program, the top-level `decomp` prompt provides an example program E_j using three sub-task functions: f_1 `:split` that *splits words in an input string*, f_2 `:str_pos` that *finds character positions in strings* and f_3 `:merge` that *concatenates characters*. In this case, we operationalize each sub-task function as a separate in-context prompt (e.g., using a standard prompting approach for `split` and `merge` on the right side), each containing a set of in-context examples that are independent of the original complex task.

In addition to the three functions described above, additional control structure is included, such as the symbolic function `foreach`, which iterates over arrays and references to previous answers such as #1. We note that such a helper function is not strictly necessary (e.g., we could directly generate "Q2': What is the first letter of Jack?" and "Q3': What is the first letter of Ryan?" instead of Q2 in the figure) and is added to reduce the manual effort needed to specify the decomposition and also reduce potential errors during decomposition. In our experiments we use two of the compositional operators defined by Khot et al. (2022) (see appendix for details), although it is capable of using all their operators (which also capture the QDMR operators from Wolfson et al. (2020)).

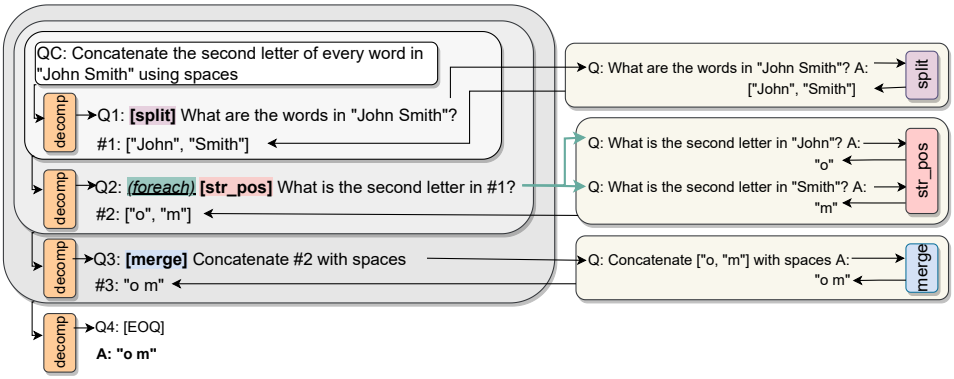


Figure 3: The inference procedure in DECOMP iteratively calls the decomposer prompt to generate the next question and sub-task at each step, given the current history of question and answers. The generated question is then routed to the assigned sub-task handler (with some handling of special operators, when needed). When the special end-of-questions `[EOQ]` marker is generated, the previous answer is returned as the final prediction.

3.2 PROMPT EXECUTION AND INFERENCE

Given a new question and a set of background in-context examples D, the inference (i.e., the program construction and execution) process is illustrated in Fig. 3. The new complex question is fed to

the decomposer prompt to get the first sub-question to be asked to the `split` prompt. With the help of our symbolic controller, the answer generated from this prompt is then appended to the decomposer prompt to get the second sub-question, $Q2$. Due to the `foreach` operator in the generated question, $Q2$ results in two questions (one for each word in the answer $\#1$) to be fed to the `str_pos` prompt. The answers are combined into an array to get the answer $\#2$. The entire decomposition history is used to generate $Q3$ and passed to the `merge` prompt to get the final answer. Since the task has been solved, the decomposition prompt produces the special end-of-sequence marker([EOQ]) and the last answer is returned as the final answer. Formally, performing inference involves finding the best answer A to a new query Q, which in the simplest form of prompting involves computing the MAP answer using the LLMs predictive distribution for A, i.e., $\hat{A} = \arg\max_A p(A \mid D, Q, \theta)$ (Dohan et al., 2022). For practicality, such computations are approximated using greedy search, which we pursue in our experiments.

3.3 DECOMP CAPABILITIES

Next we describe three types of few-shot prompting solutions that our approach enables.

Hierarchical Decomposition Certain sub-tasks, even when given many examples, are not solvable with few-shot prompting. E.g., we found identifying the k^{th} letter of a string to be challenging for the GPT3 `text-davinci-002` model. In such a scenario, we can decompose the sub-task prompt further, to first identify the letters and their position and then select the k^{th} element of this array (see Fig. 4). We can also re-use existing sub-task prompts in our framework. E.g., the `split` prompt can be reused since it was developed for the general task of splitting strings.[2]

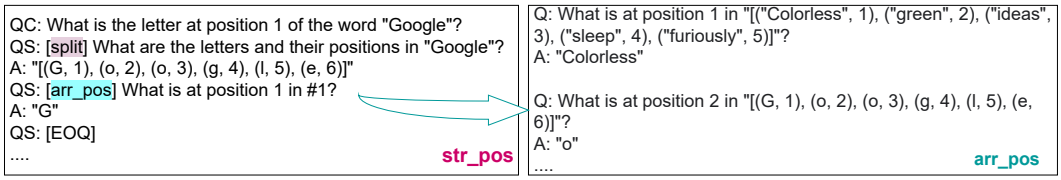


Figure 4: Since identifying the k^{th} character is challenging for GPT3 `davinci-002` model, we further decompose it into two simpler sub-tasks: split the word into its letters (using the shared sub-task `split`) and then return the k^{th} item of this list using the `arr_pos` prompt.

Recursive Decomposition Some problems can be naturally broken down into one or more smaller problems of the same form. Recursive algorithms such as merge sort use this idea to solve large problems efficiently, using a succinctly described method. We apply this same principle in DECOMP by allowing the decomposer prompt to recursively call itself, as shown in Fig. 5 for the task of list reversal. By using recursion, we are able to generalize any base prompting approach (CoT in this figure) to much longer lists by breaking the input into smaller and smaller lists till we reach a list length where the model is highly accurate. Such recursive approaches can not be described by current methods such as CoT and standard prompting. Least-to-most prompting (Zhou et al., 2022) also proposes a similar solution but differs in two key aspects (a) it has to identify all the sub-problems in one-shot instead of our iterative top-down decomposition (b) it has to learn to identify the relevant answers from the previous solutions which we get for free from our decomposition.

External API Calls In certain cases, the sub-tasks may not be feasible to solve using only a LLM. E.g., retrieving knowledge from a KB or large corpus. Such sub-tasks, however, can be easily solved using existing systems such as retrieving documents using an Elasticsearch index or webpages using Google search Lazaridou et al. (2022). Fig. 6 shows how DECOMP can easily use such a system to retrieve the relevant documents and answer a single-hop open-domain question.

[2]Appendix G contains the complete `split` prompt which has examples for questions such as "Q: What are the letters and their positions in "Mathison"?".

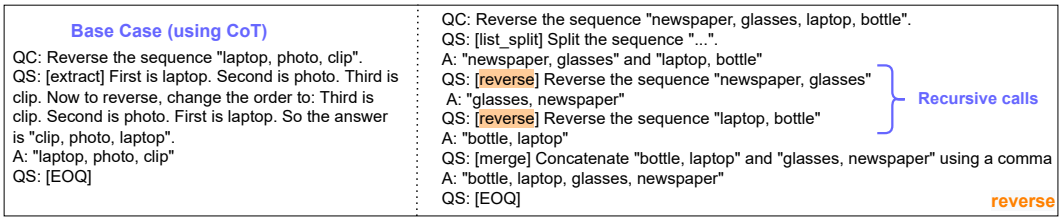


Figure 5: Sample prompt for recursive decomposition for reversing lists. Each list is split into two halves and each half is reversed and concatenated in the reverse order. We can recursively split a list till we hit the base case (lists of length 3 here) where existing approaches such as CoT are accurate.

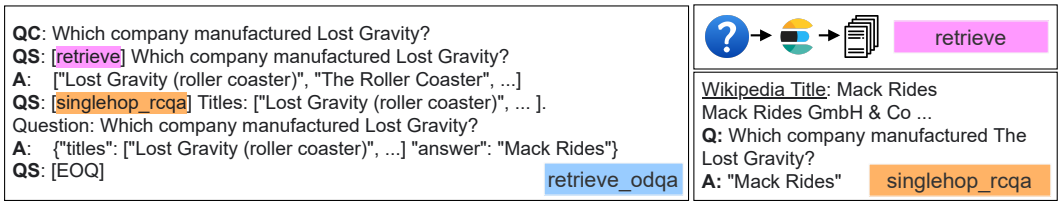


Figure 6: A Decomposed Prompt to answer open-domain questions using Elasticsearch-based retrieval. Full usage of this prompt for open-domain multihop questions is given in Figure 11.

4 Case Studies

We showcase DecomP's strengths through four tasks; two symbolic manipulation tasks similar to those investigated in Wei et al. (2022) and two existing textual multi-hop reasoning tasks. Unless specified, we use `text-davinci-002` InstructGPT3 model Ouyang et al. (2022) as the LLM and report the Exact Match (EM) numbers, following prior work. For order-independent list answers, we evaluate set equality as EM. To show the value of decomposed prompting, we compare our approach to CoT rather than each specific decomposition structure used in prior work. The complete prompts for all our tasks are provided in App. G

4.1 k^{th} letter concatenation (Hierarchical Decomposition)

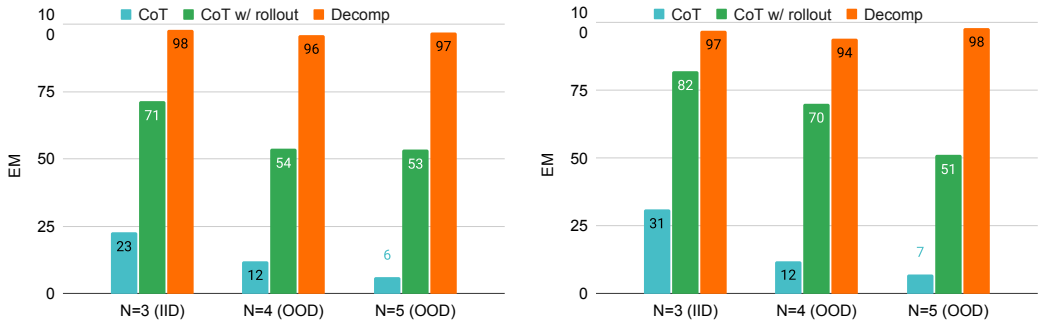
We compare DecomP to CoT prompting for concatenating letters at the k^{th} position. All prompts contain examples of concatenating letters in position 1, 4, and last position of strings with 3 words. We create three different prompts for all our baselines and present the average to account for variance due to the choice of examples following Perez et al. (2021). We use the `decomp`, `split`, `str_pos` (further decomposed as shown in Fig. 4), and `merge` prompts for decomposition prompting. We adapt the CoT for last letter concatenation from prior work (Wei et al., 2022) for this task as shown below. In addition, we consider a *rolled out* version of our decomposition prompts in terms of a CoT, i.e., we describe the entire decomposition process (identify words, split each word into letters, take k^{th} letter and concatenate) as a single CoT. e.g, for the question "Take the letters at position 4 of the words in "Herbert Alexander Simon" and concatenate them using a space.", we use the CoT:

Chain-Of-Thought	**Chain-Of-Thought (rolled out)**
The letter at position 4 of "Herbert" is "b". The letter at position 4 of "Alexander" is "x". The letter at position 4 of "Simon" is "o". Concatenating "b", "x", "o" using a space leads to "b x o". So, "Herbert Alexander Simon" outputs "b x o". ...	The words in "Herbert Alexander Simon" are "Herbert", "Alexander", and "Simon". The letters and their positions in "Herbert" are "[(H, 1), (e, 2), (r, 3), (b, 4), (e, 5), (r, 6), (t, 7)]". The letter at position 4 in this sequence is "b". $\cdots$ outputs "b x o". ...

We compare these three prompting techniques on 4 datasets to evaluate generalization along 3 axes: (1) unobserved letter position $k = 3$[3] (2) longer inputs, #words=4 and 5 (3) new delimeter "semicolon". The words in the test examples come from a list of most popular first and last names.[4] All evaluation datasets have 100 examples. See Fig. 7 for results.

[3]Note that none of the sub-task prompts contain examples for this position

[4]`forebears.io/earth/forenames` and `forebears.io/earth/surnames`



(a) Space (b) Semi-Colon

Figure 7: EM Results on the k^{th} letter concatenation task (k=3) with different values for N, the number of words in the input. DECOMP always outperforms and generalizes better than CoT.

DECOMP outperforms chain-of-thought reasoning, even when the chain-of-thought uses the same reasoning procedure as the rolled out decomposition. This shows that the separate prompts are more effective at teaching hard sub-tasks than a single CoT prompt.

DECOMP generalizes perfectly to longer sequences. As the length of the input sequence increases, our approach continues to achieve close to 100% accuracy on this task. The CoT-based approaches drop noticeably in their scores with longer input lengths, widening the performance gap.

4.2 LIST REVERSAL (RECURSIVE DECOMPOSITION)

We use the task of reversing lists of words[5]to show how recursive DECOMP enables length generalization. We adapt the relevant CoT prompt from Wei et al. (2022), and integrate it in a decomposed prompt. As a control, we also compare to a CoT version w/ rollout of our decomposed prompt. All prompts contain the same 3 examples of reversing word sequences with 3-5 items. We evaluate all prompts for generalization to 4, 6, 8, and 10-item sequences. Here we use `davinci-001` to show that DECOMP enables a weaker model approach `davinci-002`'s performance (which does solve this task). We use the strategy from Fig. 5 and provide our prompts in App. G. Figure 8 shows the results of the prompting strategies on different input lengths.

DECOMP improves the length generalization of few-shot prompting. While our base CoT prompt does not generalize at all to longer sequences, our approach can recursively decompose the problem and achieve better length generalization.

CoT w/ rollout fails. The CoT version of our decomposition strategy fails because the unrolled prompt becomes too long and convoluted without the ability to abstract away sub-modules.

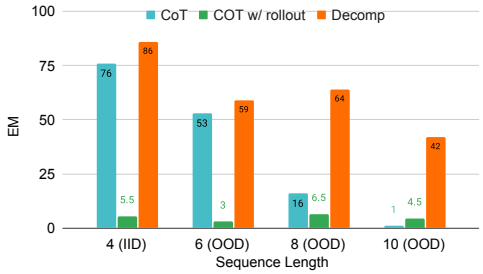


Figure 8: EM results on reversing sequences. Incorporating CoT in DECOMP greatly increases the ability of the model to generalize to new sequence lengths.

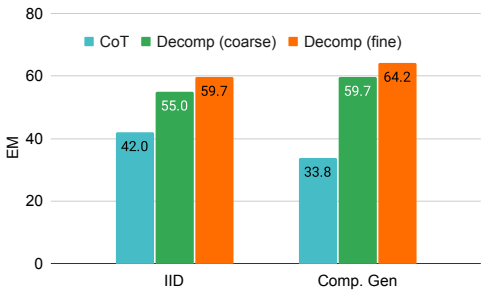


Figure 9: EM results on the CommaQA-E datasets. DECOMP always outperforms CoT, with fine-grained marginally out-performing coarse-grained decomposition.

[5]We use the vocabulary from Wei et al. (2022): `https://www.vocabulary.com/lists/189583`

4.3 LONG-CONTEXT QUESTION ANSWERING

We next evaluate on the CommaQA-E dataset (Khot et al., 2022) under the reading comprehension setting. The dataset consists of synthetically generated entities (e.g. Erowid award), facts ("Wetherality was an actor in the movie Dewbar.") and multi-hop questions (e.g., "What awards have the actors of the Erowid winning movies received?"). Due to the presence of many distractors and, as a result, longer context, this dataset has been shown to be hard for standard LMs even when fine-tuned.

To fit these questions within GPT3's context limit (2049 tokens), we generate a smaller version of the CommaQA-E dataset and of the compositional generalization split such that we can fit at least four examples in the context for CoT prompts. For the question "What awards have movies produced by people born in 1910 won?", the CoT is: *The people born in 1910 were Muntaril and Teeplemole. Teeplemole produced the movies Featsaw and Zalate. Muntaril produced the movie Premercy. Featsaw was awarded the Zorgion award. Premercy was awarded the Chowwurst award. Zalate was awarded the Hallowcock award. So the answer is ["Zorgion", "Chowwurst", "Hallowcock"].*

```
What awards have movies produced by people born in 1910 won?
QS: [qa] Who were born in the year 1910?
A: ["Teeplemole", "Muntaril"]
QS: (foreach_merge) [qa] For which movies was #1 the producer?
A: ["Featsaw", "Zalate", "Premercy"]
QS: (foreach_merge) [qa] Which awards were given to #2?
A: ["Zorgion", "Chowwurst", "Hallowcock"]
QS: [EOQ]
...                                                    coarse
```

```
What awards have movies produced by people born in 1910 won?
QS: [simp_qa] Who were born in the year 1910?
A: ["Teeplemole", "Muntaril"]
QS: (foreach_merge) [pos_qa] For which movies was #1 the producer?
A: ["Featsaw", "Zalate", "Premercy"]
QS: (foreach_merge) [aw_qa] Which awards were given to #2?
A: ["Zorgion", "Chowwurst", "Hallowcock"]
QS: [EOQ]
...                                                    fine
```

Figure 10: Sample prompts used for the CommaQA dataset. On the left, the coarse-grained decomposition defines a single QA sub-task with all single-hop questions being delegated to a single sub-task handler. On the right, the fine-grained decomposition assigns questions to three different sub-tasks (see App. G for their prompts) depending on the question type. This allows us to provide more examples for each question type allowing the model to learn the sub-task more effectively.

For DECOMP, we can separate the task of decomposition (independent of the context) from the sub-tasks of single-hop question answering. As shown in Fig. 10, we provide examples of the context-independent decomposition in the decomposer prompt and use the separate sub-task prompts to teach the QA skill over the given context. Additionally, we can choose the granularity of decomposition to trade off human effort for increased accuracy. For example, we could have single QA prompt to handle all the questions or create QA prompts for different classes of questions. In our experiments, each sub-task prompt contains 8 QA examples (2 questions/para) (see App. G for the prompts). We evaluate three different prompts and report the average results in Fig. 9.

DECOMP is more accurate than CoT on CommaQA. Irrespective of the granularity of decomposition or the evaluate split, our approach outperforms CoT on this task.

Finer grained decomposition can help improve performance. Our fine-grained decomposition can provide more examples for each class of questions resulting in increased single-hop QA performance and as a result increased performance on CommaQA.

DECOMP also generalizes to new compositions. Compositional generalization split of CommaQA uses the same relations as the training set but with new compositions. While CoT has a drop in score, both decomposition-based approaches even have slightly better scores (the subset of relations used in Comp. Gen split are easier for our specialized QA models).

```
QC: In what country was Lost Gravity manufactured?
QS: [retrieve_odqa] Which company manufactured Lost Gravity?
A:  {"titles": ["Lost Gravity (roller coaster)" ... ], "answer": ["Mack Rides"]}
QS: [retrieve_odqa] The company Mack Rides is from which country?
A:  {"titles": ["Mack Rides", ...], "answer": ["Germany"]}
QS: [multihop_rcqa] Titles: ["Lost Gravity (roller coaster)" ..., "Mack Rides" ...].
Question: In what country was Lost Gravity manufactured?      decomposer
```

```
Wikipedia Title: Lost Gravity (roller ... )
Lost Gravity is a steel roller coaster ...
Wikipedia Title: Mack Rides
Mack Rides GmbH & Co ...
    ...
Q: In what country was Lost Gravity manufactured?
A: "Germany"                                          multihop_rcqa
```

Figure 11: The prompt used to answer open-domain multihop questions using Elasticsearch-based retrieval. The `retrieve_odqa` prompt is given in Figure 6.

4.4 OPEN-DOMAIN QUESTION ANSWERING

Next, we demonstrate the ability of our approach to integrate external API calls on the task of open-domain multihop question answering. We evaluate our approach on the HotpotQA dataset under the `fullwiki` setting that requires relevant paragraphs to be retrieved from a Wikipedia corpus. We use the code-davinci-002 model here since it can fit the much longer contexts needed here.

Fig. 11 shows the decomposition prompt we use. The decomposer generates (singlehop) sub-questions and delegates them to `retrieve_odqa` (described in Fig. 6). As we showed earlier, this module retrieves relevant documents then uses an RC model to answer. `retrieve_odqa` returns both the answer and the documents, allowing subsequent sub-questions to use the answers (e.g. "Mack Rides") and the `multihop_rcqa` model to use the documents.

We implement the final `multihop_rcqa` model in two ways: (i) prompting the model to generate the answer directly (**Decomp-Ctxt QA**) as shown in Fig. 11 and (ii) prompting the model to generate the Chain-of-Thought (**Decomp-Ctxt CoT QA**) (e.g. "The Lost Gravity was manufactured by Mack Rides. Mack Rides is a German company. So the answer is: Germany."). We compare our approach against two baselines: **A. No Context (No-Ctxt),** A closed-book setting baseline where the model must rely only on its parametric knowledge. **B. NoDecomp Context (NoDecomp-Ctxt),** A simple retrieval baseline where we retrieve K (set to 8) paragraphs using the multi-hop question as the input and use that as context. In both these baselines, we consider the QA and CoT QA variants.

We manually annotate CoTs and decompositions for 25 training set questions, and sample 3 prompts of 15 questions each for all approaches. The detailed prompts are given in the Appendix G. We evaluate on the subset of bridge questions from our held-out dev questions (248/300) i.e. questions where answer for the first sub-question is needed for the second sub-question (as in our example). On the comparison questions in HotpotQA (e.g. "Who is older Jeremy Horn or Renato Sobral ?"), the relevant entities are clearly mentioned in the question and do not necessitate multi-step retrieval.[6]

We also evaluate our approach on 2WikiMultihopQA Ho et al. (2020) and analyse how the model performance changes with increasing model sizes. The results are provided in the Appendix A.

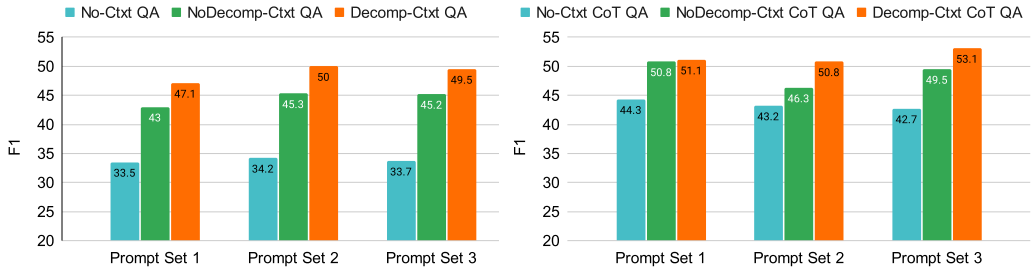


Figure 12: Answer F1[7] for comparison questions on open-domain setting of HotpotQA for three randomly sampled sets of prompt examples. (Left) Standard (non-CoT) variants (Right) CoT variants. Decomp-Ctxt models (ours) significantly outperform the corresponding No-Ctxt models, and also our strong baselines of NoDecomp-Ctxt models.

Results are presented in Fig. 12. The NoDecomp-Ctxt models perform significantly better than No-Ctxt models in both QA (i.e. standard prompting) and CoT variants showing that external knowledge can be leveraged to improve standard and CoT based open-domain mulithop QA. Furthermore, our Decomp-Ctxt models are yet better at leveraging external knowledge than a strong retrieval baseline (NoDecomp-Ctxt), regardless of whether we leverage CoT jointly with our approach or not.

[6] We did evaluate our approach on 'comparison' questions and, as expected, our performance is comparable to the NoDecomp-Ctxt baselines.

[7] Answer F1 is computed by treating prediction and ground truth answer as bags of tokens and computing their precision and recall Rajpurkar et al. (2016). See HotpotQA Yang et al. (2018) for details.

5 CONCLUSION

We proposed a new approach, Decomposed Prompting, to solve complex tasks using few-shot prompts, by decomposing them into a prompting program built out of simpler sub-tasks. Drawing inspiration from software libraries, our decomposer and shared sub-tasks are designed in a modular fashion: they use their own few-shot prompts, allowing one to independently optimize each prompt, decompose a sub-task further if necessary, or even seamlessly replace it with a symbolic system. We show that Decomposed Prompting outperforms prior work on four different tasks and generalization settings, establishing it as an effective few-shot paradigm for solving complex tasks.

REFERENCES

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 39–48, 2016.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Jonathon Cai, Richard Shin, and Dawn Song. Making neural programming architectures generalize via recursion. *Proceedings of ICLR*, 2017.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

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

Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*, 2022.

David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A Saurous, Jascha Sohl-dickstein, et al. Language model cascades. *arXiv preprint arXiv:2207.10342*, 2022.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. Neural module networks for reasoning over text. In *ICLR*, 2020.

Xanh Ho, A. Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *COLING*, 2020.

Yichen Jiang and Mohit Bansal. Self-assembling modular networks for interpretable multi-hop reasoning. *Proceedings of EMNLP*, 2019.

Jaehun Jung, Lianhui Qin, Sean Welleck, Faeze Brahman, Chandra Bhagavatula, Ronan Le Bras, and Yejin Choi. Maieutic prompting: Logically consistent reasoning with recursive explanations. *arXiv preprint arXiv:2205.11822*, 2022.

Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Text modular networks: Learning to decompose tasks in the language of existing models. In *NAACL*, 2021.

Tushar Khot, Kyle Richardson, Daniel Khashabi, and Ashish Sabharwal. Hey AI, can you solve complex tasks by talking to agents? In *Findings of ACL*, 2022.

Angeliki Lazaridou, Elena Gribovskaya, Wojciech Stokowiec, and Nikolai Grigorev. Internet-augmented language models through few-shot prompting for open-domain question answering. *ArXiv*, abs/2203.05115, 2022.

Yoav Levine, Itay Dalmedigos, Ori Ram, Yoel Zeldes, Daniel Jannai, Dor Muhlgay, Yoni Osin, Opher Lieber, Barak Lenz, Shai Shalev-Shwartz, et al. Standing on the shoulders of giant frozen language models. *arXiv preprint arXiv:2204.10019*, 2022.

Ana Marasović, Iz Beltagy, Doug Downey, and Matthew E Peters. Few-shot self-rationalization with natural language prompts. *arXiv preprint arXiv:2111.08284*, 2021.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. In *ACL*, 2019.

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. Neural programmer: Inducing latent programs with gradient descent. *Proceedings of ICLR*, 2016.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *ArXiv*, abs/2112.00114, 2021.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke E. Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Francis Christiano, Jan Leike, and Ryan J. Lowe. Training language models to follow instructions with human feedback. *ArXiv*, abs/2203.02155, 2022.

Ethan Perez, Patrick Lewis, Wen-tau Yih, Kyunghyun Cho, and Douwe Kiela. Unsupervised question decomposition for question answering. *Proceedings of EMNLP*, 2020.

Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. In *NeurIPS*, 2021.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100, 000+ questions for machine comprehension of text. In *EMNLP*, 2016.

Scott Reed and Nando De Freitas. Neural programmer-interpreters. *Proceedings of ICLR*, 2016.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1202. URL https://aclanthology.org/D15-1202.

Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *NAACL*, 2018.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171, 2022a.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Rationale-augmented ensembles in language models. *arXiv preprint arXiv:2207.00747*, 2022b.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022c.

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

Tomer Wolfson, Mor Geva, Ankit Gupta, Matt Gardner, Yoav Goldberg, Daniel Deutch, and Jonathan Berant. Break it down: A question understanding benchmark. *TACL*, 2020.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, 2018.

Eric Zelikman, Yuhuai Wu, and Noah D Goodman. Star: Bootstrapping reasoning with reasoning. *arXiv preprint arXiv:2203.14465*, 2022.

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

A 2WIKIMULTIHOPQA

We also evaluate DecomP for open-domain QA on 2WikiMultihopQA Ho et al. (2020). This dataset is reading-comprehension multihop QA dataset created by composing entities and relations on WikiData. So we first convert it to an open-domain setting by taking a union of all the paragraphs present in the dataset and treating as corpus. In all it has 430225 paragraphs. We manually annotate 20 questions with chain-of-thought prompts an DecomP prompts and manually sample 15 of them which are used as demonstrations.

To further analyse the effect of model-scale on our method, we also experiment with different sized Flan-T5 models Chung et al. (2022) (large, xl, xxl), which have been recently shown to exhibit chain-of-thought reasoning. For DecomP approach, we still use GPT3 (text-davinci-002) for decomposition as Flan variations could not generate syntactically correct decompositions. The results are show in the Figure 13.

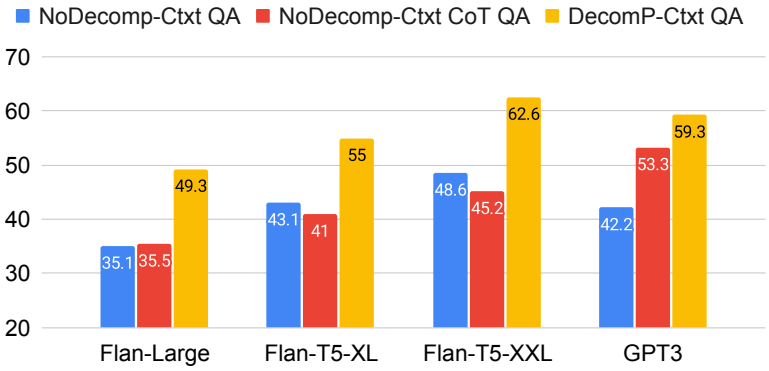


Figure 13: Results on open-domain 2WikiMultihopQA using models of different sizes. DecomP works better than NoDecomp-Ctxt QA and NoDecomp-Ctxt CoT QA for all model sizes.

B MATH QA

We apply Decomposed Prompting to two math QA datasets: GSM8K Cobbe et al. (2021) and MultiArith Roy & Roth (2015). For Chain-of-thought, we used the original prompts for math reasoning Wei et al. (2022). For example:

> Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today? A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6. The answer is 6.

Most CoT systems Wei et al. (2022); Wang et al. (2022a) rely on extracting the answer by finding the number following "answer is". However, this may not always be accurate. For example, the following CoT would be unanswerable by relying on simple patterns.

> Parker chews 4 pieces of gum a day. There are 15 pieces of gum in a pack. So he will need 4 * 30 / 15 = 8 packs of gum to last him 30 days.

Rather than relying on patterns with limited generalization, we can use a language model to extract the answer more reliably. Specifically, we use Decomposed Prompting to decompose the task into first identifying the chain-of-thought reasoning and then using a second GPT3-based sub-module to extract the answer from the CoT. We show examples of our prompts here (full prompt in App. G):

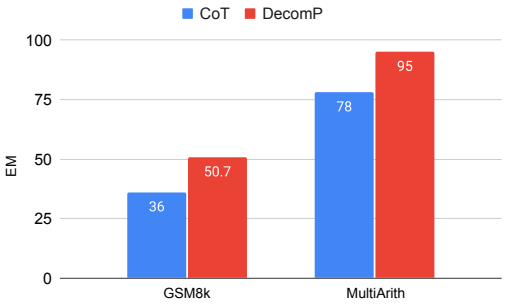


Figure 14: Our simple decomposition results in 14-17 pts on two MathQA datasets: GSM8k and MultiArith.

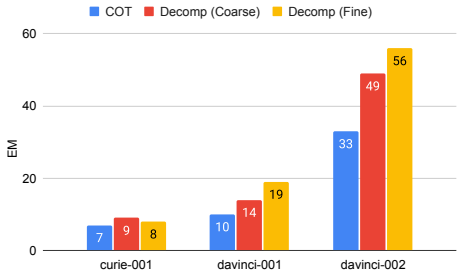


Figure 15: As the models become weaker (davinci-001) and smaller (curie-001), the performance of all the models drop. DECOMP still outperforms CoT till the performance reaches close to zero with curie.

Decomposition Prompt
QC: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
QS: [cot] There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6 trees planted.
QS: [gpt_ans] There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6 trees planted.
A: 6
QS: [EOQ]

gpt_ans
Q: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 - 15 = 6 trees planted.
A: 6

We present our results in Fig. 14. On the GSM8K data set[8], we outperform CoT by 14 points. On the MultiArith dataset[9], we achieve a 17 pt improvement compare to CoT. While this is a simple change, it illustrates the possibility of using DECOMP for other complex answer types, e.g. non-extractive answer generation from chain-of-thoughts.

C EFFECT OF SCALE ON COMMAQA

We evaluate text-curie-001, text-davinci-001 and text-davinci-002 on the CommAQA dataset. Since the curie-001 and davinci-001 have a smaller context window size, we further reduced our prompts to fit within their context windows (2048 tokens). As shown in Fig. 15, both CoT and DECOMP are effected by the model size.

D RESULTS ON ALL PROMPTS

D.1 PER-PROMPT RESULT ON LETTER CONCATENATION

We present the results of the letter concatenation task (with space delimiter) for different values of N in Fig. 16. Our results are stable across the different prompts (P1, P2 and P3) and always outperform CoT. We also compare our system to least-to-most prompting Zhou et al. (2022) (prompts in App. G). Since least-to-most prompting can not decompose the k^{th} letter extraction sub-task further, its performance is similar to CoT and much worse than DECOMP.

[8]We randomly sample 300 examples from the test set due to costs with API usage
[9]We randomly sample 200 examples from the test set due to costs with API usage

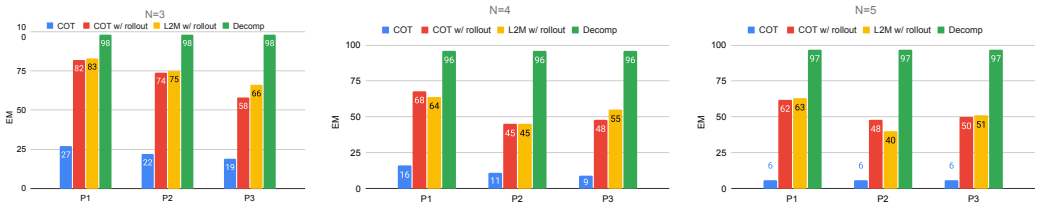
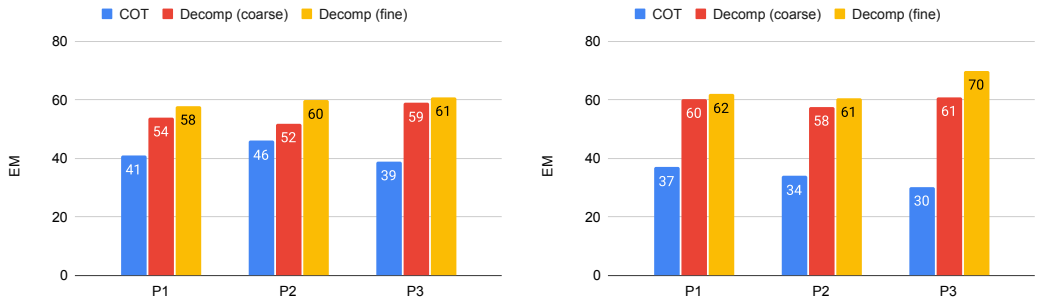


Figure 16: Across all values of N and different prompts (P1, P2 and P3), DECOMP outperform chain-of-thought reasoning and even least-to-most promptimg.

D.2 PER-PROMPT RESULTS OM COMMAQA

We present the results of all the prompts on the CommAQA dataset in Fig. 17



(a) Test Set

(b) Comp. Gen. Set

Figure 17: Results of different prompts on the CommAQA dataset.

E EFFECT OF DECOMPOSITION SCHEME

To evaluate the effect of the decomposition scheme, we experiment with two other simple decomposition structures for the letter concatenation and reversal tasks.

For letter concatenation, we consider an alternate scheme where we use GPT3 to generate each question rather than loop over the answers, i.e.,

Replace
A: ["Augusta", Ada", King"]
QS: (foreach) [str_position] What is the last letter in "#1"?
with
QS: [str_position] What is the last letter in "Augusta"?
A: "a"
QS: [str_position] What is the last letter in "Ada"?
A: "a"
QS: [str_position] What is the last letter in "King"?
A: "g"

By using the decomposer prompt model to generate these sub-questions, we can be more robust to formatting issues in the output answers, but at the cost of potential precision or recall mistakes in generating these questions.

For list reversal, instead of splitting into halves, we take the tail of the list, reverse it and then concatenate it to the head. i.e. reverse(list) = reverse(list[1:]) + list[0]. This requires more GPT3 calls (O(n)) compared to the original approach of splitting the list into halves.

In both these cases, we noticed that the performance did not drop as shown in Fig. 18 and Fig. 19. On the letter concatenation task, the results were exactly the same. The new reversal decomposition

schema was actually stronger on longer inputs at the cost of more calls to GPT3 (O(ln(n)) using binary splits vs O(n) one element at a time). Both these decomposition schemes are still better than CoT.

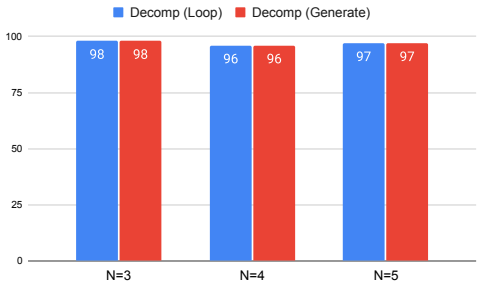


Figure 18: Both decomposition schemes for the letter concatenation task have the same scores.

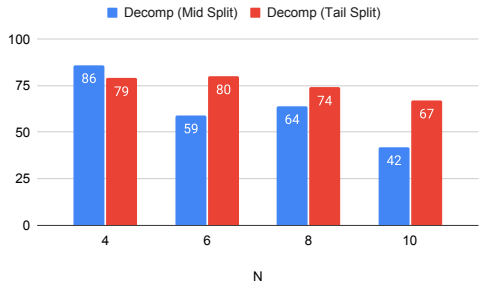


Figure 19: Recursively reversing the tail of a list is more stable at longer lengths but comes at the cost of more calls to GPT3.

F Errors

F.1 Letter Concatenation

F.1.1 DecomP

Q: Take the letters at position 3 of the words in ”Nancy Samina Abbas Caudhari Bano” and concatenate them using a space.
A: n m b u n
Prediction: c m b u n
Error: Incorrect letter extraction (Sub-task)
What is at position 3 in ”[(N, 1), (a, 2), (n, 3), (c, 4), (y, 5)]”? ⇒ ”c”

Q: Take the letters at position 3 of the words in ”Orlando Stephen Cho Teixeira Pierre” and concatenate them using a space.
A: l e o i e
Prediction: leoie
Error: Incorrect concatenation (Sub-task)
Concatenate [”l”, ”e”, ”o”, ”i”, ”e”] using a space. ⇒ ”leoie”

F.1.2 CoT w/ rollout

Q: Take the letters at position 3 of the words in ”Sheila Nicolas Verma Sha Sousa” and concatenate them using a space.
A: e c r a u
Pred: i c r a u
Error: Incorrect letter extraction
...The letters and their positions in ”Sheila” are ”[(S, 1), (h, 2), (e, 3), (i, 4), (l, 5), (a, 6)]”. The letter at position 3 in this sequence is ”i”...

Q:Take the letters at position 3 of the words in ”Shobha Kailash Nakamura Peter Benitez” and concatenate them using a space.
A: o i k t n
Pred: o l k t i
Error: Incorrect letter extraction
...”Benitez” are ”[(B, 1), (e, 2), (n, 3), (i, 4), (t, 5), (e, 6), (z, 7)]”. The letter at position 3 in this sequence is ”i”...

F.2 COMMAQA

Similarly in CommaQA, the errors are mostly due to sub-task errors, which in this dataset correspond to answering single-hop questions. CoT also makes the same types of errors but they are more frequent since this QA sub-task can not be delegated to a specialized prompt in CoT. Since all errors are of this type, we show only one example here.

Q: What awards have movies written by people born in 1933 won?
A: ["Hydrallium", "Pompasole"]
Pred: ["Pompasole"]
Error: Incorrect sub-question answer
Sub-Q:What movies has Haldron written?
Sub-A: ["Polytetrafluoromethane", "Skia", "Warpstone"]
Pred: ["Skia", "Warpstone"]

Similar sub-task errors are present in the list reversal and open-domain QA datasets.

G TASK PROMPTS

We have provided the task prompts for all the datasets for COT and our Decomposed Prompting approach.

CoT Since CoT methods also perform 2-step reasoning: first generate the chain-of-thought and second extract the answer from the CoT, we use the same decomposition-based framework for COT baselines too. For example, consider the following example in our COT prompt:

QC: Take the letters at position 1 of the words in "Alan Mathison Turing" and concatenate them using a space.
QS: [extract] The letter at position 1 of "Alan" is "A". The letter at position 1 of "Mathison" is "M". The letter at position 1 of "Turing" is "T". Concatenating "A", "M", "T" using a space leads to "A M T". So, "Alan Mathison Turing" outputs "A M T".
A: "A M T"
QS: [EOQ]

GPT3 generates the chain-of-thought during the "decomposition" step and a regex-based answer extractor `extract('.* outputs "(.*)"\.')` then takes this CoT and generates the answer. In some cases, the module name is skipped in the prompt (the CoT is sent to the extractor by default).

Operators In this work, we use the same operators as defined by Khot et al.. Their `select` operator is just the basic operator that replaces references to an answer index with its answer. When not specified, `select` is assumed to be the default operator. In addition, we consider two operators in our experiments: `project_values` and `project_values_flat_unique`.

- `project_values`: This operator takes a list answer $\#i = X$ and iterates over it to generate new questions by replacing mentions of $\#i$ i.e. Q = [q.replace(#i, x) for x $\in$ X]. The answer to each question is simply concatenated to get the final answer i.e. A = [model(q) for q $\in$ Q]. We refer to this as `foreach` for simplicity in the main text.
- `project_values_flat_unique`: This operator performs the same steps as `project_values` but then additionally flattens the list and only returns the unique entities in the flattened list. We refer to this as `foreach_merge` in the main text for simplicity.

G.1 LETTER CONCATENTATION

We show one of the prompts used for experiments here. The entire set of prompts is provided as supplemetary material.

G.1.1 DECOMPOSED PROMPTING

`decomp`

```
QC: Take the last letters of the words in ”Augusta Ada King” and concatenate them using a space.
QS: [split] What are the words in ”Augusta Ada King”?
A: [”Augusta”, ”Ada”, ”King”]
QS: (project_values) [str_position] What is the last letter in ”#1”?
A: [”a”, ”a”, ”g”]
QS: [merge] Concatenate #2 using a space.
A: ”a a g”
QS: [EOQ]

QC: Take the letters at position 1 of the words in ”Alan Mathison Turing” and concatenate them
    using a space.
QS: [split] What are the words in ”Alan Mathison Turing”?
A: [”Alan”, ”Mathison”, ”Turing”]
QS: (project_values) [str_position] What is the letter at position 1 in ”#1”?
A: [”A”, ”M”, ”T”]
QS: [merge] Concatenate #2 using a space.
A: ”A M T”
QS: [EOQ]

QC: Take the letters at position 4 of the words in ”Herbert Alexander Simon” and concatenate
    them using a space.
QS: [split] What are the words in ”Herbert Alexander Simon”?
A: [”Herbert”, ”Alexander”, ”Simon”]
QS: (project_values) [str_position] What is the letter at position 4 in ”#1”?
A: [”b”, ”x”, ”o”]
QS: [merge] Concatenate #2 using a space.
A: ”b x o”
QS: [EOQ]
```

`split`

```
Q: What are the words in ”Alan Mathison Turing”?
A: [”Alan”, ”Mathison”, ”Turing”]

Q: What are the letters in ”Alan”?
A: [”A”, ”l”, ”a”, ”n”]

Q: What are the letters and their positions in ”Mathison”?
A: ”[(M, 1), (a, 2), (t, 3), (h, 4), (i, 5), (s, 6), (o, 7), (n, 8)]”

Q: What are the words and their positions in ”Herbert Alexander Simon”?
A: ”[(Herbert, 1), (Alexander, 2), (Simon, 3)]”
```

`str_position`

```
QC: What is the letter at position 1 of the word ”Augusta”?
QS: (select) [split] What are the letters and their positions in ”Augusta”?
A: ”[(A, 1), (u, 2), (g, 3), (u, 4), (s, 5), (t, 6), (a, 7)]”
QS: (select) [arr_position] What is at position 1 in #1?
A: ”A”
QS: [EOQ]

QC: What is the last letter of the word ”Mathison”?
QS: (select) [split] What are the letters and their positions in ”Mathison”?
A: ”[(M, 1), (a, 2), (t, 3), (h, 4), (i, 5), (s, 6), (o, 7), (n, 8)]”
QS: (select) [arr_position] What is the last letter in #1?
A: ”n”
```

QS: [EOQ]

QC: What is the word at the position 4 in ”Colorless green ideas sleep furiously”?
QS: (select) [split] What are the words and their positions in ”Colorless green ideas sleep furiously ”?
A: ”[(Colorless, 1), (green, 2), (ideas, 3), (sleep, 4), (furiously, 5)]”
QS: (select) [arr_position] What is at the position 4 in #1?
A: ”sleep”
QS: [EOQ]

merge

Q: Concatenate [”A”, ”l”, ”a”, ”n”].
A: ”Alan”

Q: Concatenate [”b”, ”x”, ”o”] using a space.
A: ”b x o”

Q: Concatenate [”a”, ”a”, ”g”] using a comma.
A: ”a,a,g”

Q: Concatenate [”Alan”, ”Mathison”, ”Turing”] using a space.
A: ”Alan Mathison Turing”

Q: Concatenate [”Allen”, ”Institute”].
A: ”AllenInstitute”

arr_position

Q: What is at position 4 in ”[(”Colorless”, 1), (”green”, 2), (”ideas”, 3), (”sleep”, 4), (”furiously”, 5)]”?
A: ”sleep”

Q: What is at position 1 in ”[(M, 1), (a, 2), (t, 3), (h, 4), (i, 5), (s, 6), (o, 7), (n, 8)]”?
A: ”M”

Q: What is at the last position in ”[(A, 1), (u, 2), (g, 3), (u, 4), (s, 5), (t, 6), (a, 7)]”?
A: ”a”

Q: What is at position 1 in ”[(Herbert, 1), (Alexander, 2), (Simon, 3)]”?
A: ”Herbert”

Q: What is at last position in ”[(Allen, 1), (Institute, 2), (for, 3), (Artificial, 4), (Intelligence, 5)]”?
A: ”Intelligence”

Q: What is at position 4 in ”[(A, 1), (l, 2), (e, 3), (x, 4), (a, 5), (n, 6), (d, 7), (e, 8), (r, 9)]”?
A: ”x”

G.1.2 COT WITH ROLLOUT

COT w/ rollout

QC: Take the last letters of the words in ”Augusta Ada King” and concatenate them using a space.
QS: The words in ”Augusta Ada King” are ”Augusta”, ”Ada” and ”King”. The letters and their positions in ”Augusta” are ”[(A, 1), (u, 2), (g, 3), (u, 4), (s, 5), (t, 6), (a, 7)]”. The last letter in this sequence is ”a”. The letters and their positions in ”Ada” are ”[(A, 1), (d, 2), (a, 3)]”. The last letter in this sequence is ”a”. The letters and their positions in ”King” are ”[(K, 1), (i, 2), (n, 3), (g, 4)]”. The last letter in this sequence is ”g”. Concatenating ”a”, ”a”, ”g” using a space leads to ”a a g”. So, ”Augusta Ada King” outputs ”a a g”.
A: ”a a g”
QS: [EOQ]

QC: Take the letters at position 1 of the words in ”Alan Mathison Turing” and concatenate them using a space.
QS: The words in ”Alan Mathison Turing” are ”Alan”, ”Mathison”, and ”Turing”. The letters and their positions in ”Alan” are ”[(A, 1), (l, 2), (a, 3), (n, 4)]”. The letter at position 1 in this sequence is ”A”. The letters and their positions in ”Mathison” are [(M, 1), (a, 2), (t, 3), (h, 4), (i, 5), (s, 6), (o, 7), (n, 8)]”. The letter at position 1 in this sequence is ”M”. The letters and their positions in ”Turing” are ”[(T, 1), (u, 2), (r, 3), (i, 4), (n, 5), (g, 6)]”. The letter at position 1 in this sequence is ”T”. Concatenating ”A”, ”M”, ”T” using a space leads to ”A M T”. So, ”Alan Mathison Turing” outputs ”A M T”.
A: ”A M T”
QS: [EOQ]

QC: Take the letters at position 4 of the words in ”Herbert Alexander Simon” and concatenate them using a space.
QS: The words in ”Herbert Alexander Simon” are ”Herbert”, ”Alexander”, and ”Simon”. The letters and their positions in ”Herbert” are ”[(H, 1), (e, 2), (r, 3), (b, 4), (e, 5), (r, 6), (t, 7)]”. The letter at position 4 in this sequence is ”b”. The letters and their positions in ”Alexander” are ”[(A, 1), (l, 2), (e, 3), (x, 4), (a, 5), (n, 6), (d, 7), (e, 8), (r, 9)]”. The letter at position 4 in this sequence is ”x”. The letters and their positions in ”Simon” are ”[(S, 1), (i, 2), (m, 3), (o, 4), (n, 5)]”. The letter at position 4 in this sequence is ”o”. Concatenating ”b”, ”x”, ”o” using a space leads to ”b x o”. So, ”Herbert Alexander Simon” outputs ”b x o”.
A: ”b x o”
QS: [EOQ]

G.1.3 COT

COT

QC: Take the last letters of the words in ”Augusta Ada King” and concatenate them using a space.
QS: The last letter of ”Augusta” is ”a”. The last letter of ”Ada” is ”a”. The last letter of ”King” is ”g”. Concatenating ”a”, ”a”, ”g” using a space leads to ”a a g”. So, ”Augusta Ada King” outputs ”a a g”.
A: ”a a g”
QS: [EOQ]

QC: Take the letters at position 1 of the words in ”Alan Mathison Turing” and concatenate them using a space.
QS: The letter at position 1 of ”Alan” is ”A”. The letter at position 1 of ”Mathison” is ”M”. The letter at position 1 of ”Turing” is ”T”. Concatenating ”A”, ”M”, ”T” using a space leads to ”A M T”. So, ”Alan Mathison Turing” outputs ”A M T”.
A: ”A M T”
QS: [EOQ]

QC: Take the letters at position 4 of the words in ”Herbert Alexander Simon” and concatenate them using a space.
QS: The letter at position 4 of ”Herbert” is ”b”. The letter at position 4 of ”Alexander” is ”x”. The letter at position 4 of ”Simon” is ”o”. Concatenating ”b”, ”x”, ”o” using a space leads to ”b x o”. So, ”Herbert Alexander Simon” outputs ”b x o”.
A: ”b x o”
QS: [EOQ]

G.1.4 LEAST-TO-MOST W/ ROLLOUT

Least-to-most Decomp

QC: Take the last letters of the words in ”Augusta Ada King” and concatenate them using a space.
QS: [l2m] Take the last letters of the words in ”Augusta Ada” and concatenate them using a space.
A: The words in ”Augusta Ada” are ”Augusta” and ”Ada”. The letters and their positions in ”Augusta” are ”[(A, 1), (u, 2), (g, 3), (u, 4), (s, 5), (t, 6), (a, 7)]”. The last letter in this sequence

is ”a”. The letters and their positions in ”Ada” are ”[(A, 1), (d, 2), (a, 3)]”. The last letter in this sequence is ”a”. Concatenating ”a”, ”a” using a space leads to ”a a”. So, ”Augusta Ada” outputs ”a a”.
QS: [l2m] Take the last letters of the words in ”Augusta Ada King” and concatenate them using a space.
A: ”Augusta Ada” outputs ”a a”. The letters and their positions in ”King” are ”[(K, 1), (i, 2), (n, 3), (g, 4)]”. The last letter in this sequence is ”g”. Concatenating ”a a”, ”g” using a space leads to ”a a g”. So, ”Augusta Ada King” outputs ”a a g”.
QS: [ans_ext] So, ”Augusta Ada King” outputs ”a a g”.
A: ”a a g”
QS: [EOQ]

QC: Take the letters at position 1 of the words in ”Alan Mathison Turing” and concatenate them using a space.
QS: [l2m] Take the letters at position 1 of the words in ”Alan Mathison” and concatenate them using a space.
A: The words in ”Alan Mathison” are ”Alan” and ”Mathison”. The letters and their positions in ”Alan” are ”[(A, 1), (l, 2), (a, 3), (n, 4)]”. The letter at position 1 in this sequence is ”A”. The letters and their positions in ”Mathison” are [(M, 1), (a, 2), (t, 3), (h, 4), (i, 5), (s, 6), (o, 7), (n, 8)]”. The letter at position 1 in this sequence is ”M”. Concatenating ”A”, ”M” using a space leads to ”A M”. So, ”Alan Mathison” outputs ”A M”.
QS: [l2m] Take the letters at position 1 of the words in ”Alan Mathison Turing” and concatenate them using a space.
A: ”Alan Mathison” outputs ”A M”. The letters and their positions in ”Turing” are ”[(T, 1), (u, 2), (r, 3), (i, 4), (n, 5), (g, 6)]”. The letter at position 1 in this sequence is ”T”. Concatenating ”A M”, ”T” using a space leads to ”A M T”. So, ”Alan Mathison Turing” outputs ”A M T”.
QS: [ans_ext] So, ”Alan Mathison Turing” outputs ”A M T”.
A: ”A M T”
QS: [EOQ]

QC: Take the letters at position 4 of the words in ”Herbert Alexander Simon” and concatenate them using a space.
QS: [l2m] Take the letters at position 4 of the words in ”Herbert Alexander” and concatenate them using a space.
A: The words in ”Herbert Alexander” are ”Herbert” and ”Alexander”. The letters and their positions in ”Herbert” are ”[(H, 1), (e, 2), (r, 3), (b, 4), (e, 5), (r, 6), (t, 7)]”. The letter at position 4 in this sequence is ”b”. The letters and their positions in ”Alexander” are ”[(A, 1), (l, 2), (e, 3), (x, 4), (a, 5), (n, 6), (d, 7), (e, 8), (r, 9)]”. The letter at position 4 in this sequence is ”x”. Concatenating ”b”, ”x” using a space leads to ”b x”. So, ”Herbert Alexander” outputs ” b x”.
QS: [l2m] Take the letters at position 4 of the words in ”Herbert Alexander Simon” and concatenate them using a space.
A: ”Herbert Alexander” outputs ”b x”. The letters and their positions in ”Simon” are ”[(S, 1), (i, 2) , (m, 3), (o, 4), (n, 5)]”. The letter at position 4 in this sequence is ”o”. Concatenating ”b x”, ” o” using a space leads to ”b x o”. So, ”Herbert Alexander Simon” outputs ”b x o”.
QS: [ans_ext] So, ”Herbert Alexander Simon” outputs ”b x o”.
A: ”b x o”
QS: [EOQ]

`Least-to-most COT(l2m)`

Q: Take the last letters of the words in ”Augusta Ada” and concatenate them using a space.
A: The words in ”Augusta Ada” are ”Augusta” and ”Ada”. The letters and their positions in ” Augusta” are ”[(A, 1), (u, 2), (g, 3), (u, 4), (s, 5), (t, 6), (a, 7)]”. The last letter in this sequence is ”a”. The letters and their positions in ”Ada” are ”[(A, 1), (d, 2), (a, 3)]”. The last letter in this sequence is ”a”. Concatenating ”a”, ”a” using a space leads to ”a a”. So, ”Augusta Ada” outputs ”a a”.

Q: Take the last letters of the words in ”Augusta Ada King” and concatenate them using a space.

A: ”Augusta Ada” outputs ”a a”. The letters and their positions in ”King” are ”[(K, 1), (i, 2), (n, 3), (g, 4)]”. The last letter in this sequence is ”g”. Concatenating ”a a”, ”g” using a space leads to ”a a g”. So, ”Augusta Ada King” outputs ”a a g”.

Q: Take the letters at position 1 of the words in ”Alan Mathison” and concatenate them using a space.
A: The words in ”Alan Mathison” are ”Alan” and ”Mathison”. The letters and their positions in ”Alan” are ”[(A, 1), (l, 2), (a, 3), (n, 4)]”. The letter at position 1 in this sequence is ”A”. The letters and their positions in ”Mathison” are [(M, 1), (a, 2), (t, 3), (h, 4), (i, 5), (s, 6), (o, 7), (n, 8)]”. The letter at position 1 in this sequence is ”M”. Concatenating ”A”, ”M” using a space leads to ”A M”. So, ”Alan Mathison” outputs ”A M”.

Q: Take the letters at position 1 of the words in ”Alan Mathison Turing” and concatenate them using a space.
A: ”Alan Mathison” outputs ”A M”. The letters and their positions in ”Turing” are ”[(T, 1), (u, 2), (r, 3), (i, 4), (n, 5), (g, 6)]”. The letter at position 1 in this sequence is ”T”. Concatenating ”A M”, ”T” using a space leads to ”A M T”. So, ”Alan Mathison Turing” outputs ”A M T”.

Q: Take the letters at position 4 of the words in ”Herbert Alexander” and concatenate them using a space.
A: The words in ”Herbert Alexander” are ”Herbert” and ”Alexander”. The letters and their positions in ”Herbert” are ”[(H, 1), (e, 2), (r, 3), (b, 4), (e, 5), (r, 6), (t, 7)]”. The letter at position 4 in this sequence is ”b”. The letters and their positions in ”Alexander” are ”[(A, 1), (l, 2), (e, 3), (x, 4), (a, 5), (n, 6), (d, 7), (e, 8), (r, 9)]”. The letter at position 4 in this sequence is ”x”. Concatenating ”b”, ”x” using a space leads to ”b x”. So, ”Herbert Alexander” outputs ” b x”.

Q: Take the letters at position 4 of the words in ”Herbert Alexander Simon” and concatenate them using a space.
A: ”Herbert Alexander” outputs ”b x”. The letters and their positions in ”Simon” are ”[(S, 1), (i, 2) , (m, 3), (o, 4), (n, 5)]”. The letter at position 4 in this sequence is ”o”. Concatenating ”b x”, ” o” using a space leads to ”b x o”. So, ”Herbert Alexander Simon” outputs ”b x o”.

G.1.5 ALT DECOMP SCHEMA (GENERATE)

decomp

QC: Take the last letters of the words in ”Augusta Ada King” and concatenate them using a space.
QS: [split] What are the words in ”Augusta Ada King”?
A: [”Augusta”, ”Ada”, ”King”]
QS: [str_position] What is the last letter in ”Augusta”?
A: ”a”
QS: [str_position] What is the last letter in ”Ada”?
A: ”a”
QS: [str_position] What is the last letter in ”King”?
A: ”g”
QS: [merge] Concatenate [”a”, ”a”, ”g”] using a space.
A: ”a a g”
QS: [EOQ]

QC: Take the letters at position 1 of the words in ”Alan Mathison Turing” and concatenate them using a space.
QS: [split] What are the words in ”Alan Mathison Turing”?
A: [”Alan”, ”Mathison”, ”Turing”]
QS: [str_position] What is the letter at position 1 in ”Alan”?
A: ”A”
QS: [str_position] What is the letter at position 1 in ”Mathison”?
A: ”M”
QS: [str_position] What is the letter at position 1 in ”Turing”?
A: ”T”

QS: [merge] Concatenate [”A”, ”M”, ”T”] using a space.
A: ”A M T”
QS: [EOQ]

QC: Take the letters at position 4 of the words in ”Herbert Alexander Simon” and concatenate them using a space.
QS: [split] What are the words in ”Herbert Alexander Simon”?
A: [”Herbert”, ”Alexander”, ”Simon”]
QS: [str_position] What is the letter at position 4 in ”Herbert”?
A: ”b”
QS: [str_position] What is the letter at position 4 in ”Alexander”?
A: ”x”
QS: [str_position] What is the letter at position 4 in ”Simon”?
A: ”o”
QS: [merge] Concatenate [”b”, ”x”, ”o”] using a space.
A: ”b x o”
QS: [EOQ]

G.2 SEQUENCE REVERSAL

G.2.1 SPLIT REVERSEAL

The prompts in this section implement Algorithm 1.

Algorithm 1 A recursive reversal strategy that splits the sequence in half, reverses each half, and concatenates them. Runs in $O(\log n)$ calls to the LM where n is the number of items in the sequence.

procedure SPLITREVERSE(x) ▷ The reversal of x
  if $|x| < 4$ **then** ▷ The base case
    return $x_{|x|}, \ldots, x_1$ ▷ The reversed sequence
  else ▷ The inductive case
    $n \leftarrow |x|/2$ ▷ Half the length of x
    $\ell \leftarrow x_1, \ldots, x_n$ ▷ The first half of x
    $\ell^R \leftarrow$ SPLITREVERSE(ℓ) ▷ The reversed first half
    $r \leftarrow x_{n+1}, \ldots, x_{|x|}$ ▷ The second half of x
    $r^R \leftarrow$ SPLITREVERSE(r) ▷ The reversed second half
    return $r_1^R, \ldots, r_n^R, \ell_{n+1}^R, \ldots, \ell_{|x|}^R$ ▷ The concatenated reversed halves
  end if
end procedure

`reverse`

QC: Reverse the sequence ”driving license, button, packet, identity card, shoe”.
QS: [extract] The sequence is ”1. driving license, 2. button, 3. packet, 4. identity card, 5. shoe”. The sequence is 5 items long, which is more than the minimum length of 4, so we split it. Half of 5 is 5 / 2 = 2.5. Dropping the decimal, we get that the first half will be 2 items long, ending in ”2. button”. The first half (2 items) is ”1. driving license, 2. button”.
A: ”1. driving license, 2. button”
QS: [extract] The first half of the sequence ends with ”2. button”, so the second half starts after ”2. button” with ”3. packet”. The full sequence is 5 items long, and the first half is 2 items long, so the second half will be 5 − 2 = 3 items long. The second half of the sequence (3 items) is ”3. packet, 4. identity card, 5. shoe”.
A: ”3. packet, 4. identity card, 5. shoe”
QS: [remove_numbers] Remove the numbers from #1.
A: ”driving license, button”
QS: [remove_numbers] Remove the numbers from #2.
A: ”packet, identity card, shoe”
QS: [reverse] Reverse the sequence #3.
A: ”button, driving license”

QS: [reverse] Reverse the sequence #4.
A: ”shoe, identity card, packet”
QS: [join] #6 #5
A: ”shoe, identity card, packet, button, driving license”
QS: [EOQ]

QC: Reverse the sequence ”laptop, photo, clip”.
QS: [extract] The sequence is ”1. laptop, 2. photo, 3. clip”. The sequence is 3 items long, which is less than the minimum length of 4, so we don’t need to split it. All we need to do is reverse ” laptop, photo, clip”.
A: ”laptop, photo, clip”
QS: [cot] Reverse the sequence #1.
A: ”clip, photo, laptop”
QS: [EOQ]

QC: Reverse the sequence ”newspaper, glasses, laptop, bottle”.
QS: [extract] The sequence is ”1. newspaper, 2. glasses, 3. laptop, 4. bottle”. The sequence is 4 items long, which is equal to the minimum length of 4, so we split it. Half of 4 is 4 / 2 = 2.0. Dropping the decimal, we get that the first half will be 2 items long. The first half (2 items) of the sequence is ”1. newspaper, 2. glasses”.
A: ”1. newspaper, 2. glasses”
QS: [extract] The first half of the sequence ends with ”2. glasses”, so the second half starts after ”2. glasses” with ”3. laptop”. The full sequence is 4 items long and the first half is 2 items long, so the second half will be 4 − 2 = 2 items long, ending in ”2. glasses”. The second half of the sequence (2 items) is ”3. laptop, 4. bottle”.
A: ”3. laptop, 4. bottle”
QS: [remove_numbers] Remove the numbers from #1.
A: ”newspaper, glasses”
QS: [remove_numbers] Remove the numbers from #2.
A: ”laptop, bottle”
QS: [reverse] Reverse the sequence #3.
A: ”glasses, newspaper”
QS: [reverse] Reverse the sequence #4.
A: ”bottle, laptop”
QS: [join] #6 #5
A: ”bottle, laptop, glasses, newspaper”
QS: [EOQ]

`remove_numbers`

Q: Remove the numbers from ”4. bottle, 3. laptop, 2. glasses, 1. newspaper”.
A: ”bottle, laptop, glasses, newspaper”

Q: Remove the numbers from ”1. identity card, 2. packet, 3. button”.
A: ”identity card, packet, button”

Q: Remove the numbers from ”1. player, 2. passport, 3. umbrella, 4. radio”.
A: ”player, passport, umbrella, radio”

`join`

Q: ”bottle, laptop” ”glasses, newspaper”.
A: ”bottle, laptop, glasses, newspaper”

Q: ”identity card, packet, button” ”magazine, notebook, glasses”.
A: ”identity card, packet, button, magazine, notebook, glasses”

Q: ”passport, umbrella, radio, mobile phone, photo” ”player”.
A: ”passport, umbrella, radio, mobile phone, photo, player”

Q: ”mirror, case” ”toothbrush, alarm clock”.
A: ”mirror, case, toothbrush, alarm clock”

Q: ”light bulb, clip, umbrella” ”driving licence, watch”.
A: ”light bulb, clip, umbrella, driving licence, watch”

cot

QC: Reverse the sequence ”newspaper, glasses, laptop, bottle”.
QS: First is newspaper. Second is glasses. Third is laptop. Fourth is bottle. Now to reverse, change the order to: Fourth is bottle. Third is laptop. Second is glasses. First is newspaper. So the answer is ”bottle, laptop, glasses, newspaper”.
A: ”bottle, laptop, glasses, newspaper”

QC: Reverse the sequence ”laptop, photo, clip”.
QS: First is laptop. Second is photo. Third is clip. Now to reverse, change the order to: Third is clip . Second is photo. First is laptop. So the answer is ”clip, photo, laptop”.
A: ”clip, photo, laptop”

QC: Reverse the sequence ”driving license, button, packet, identity card, pineapple”.
QS: First is driving license. Second is button. Third is packet. Fourth is identity card. Fifth is pineapple. Now to reverse, change the order to: Fifth is pineapple. Fourth is identity card. Third is packet. Second is button. First is driving license. So the answer is ”pineapple, identity card, packet, button, driving license”.
A: ”pineapple, identity card, packet, button, driving license”

unrolled_decomp

QC: Reverse the sequence ”driving license, button, packet, identity card, shoe”.
QS: The sequence is ”1. driving license, 2. button, 3. packet, 4. identity card, 5. shoe”. The sequence is 5 items long, which is more than the minimum length of 4, so we split it. Half of 5 is 5 / 2 = 2.5. Dropping the decimal, we get that the first half will be 2 items long, ending in ”2. button”. The first half (2 items) is ”1. driving license, 2. button”. The first half of the sequence ends with ”2. button”, so the second half starts after ”2. button” with ”3. packet”. The full sequence is 5 items long, and the first half is 2 items long, so the second half will be 5 − 2 = 3 items long. The second half of the sequence (3 items) is ”3. packet, 4. identity card, 5. shoe”. Removing the numbers from ”1. driving license, 2. button”, we get ”driving license, button”. Removing the numbers from ”3. packet, 4. identity card, 5. shoe”, we get ”packet, identity card, shoe”. Reversing the sequence ”driving license, button”, First is driving license. Second is button. Now to reverse, change the order to: Second is button. First is driving license. So the answer is ”button, driving license”. Reversing the sequence ”packet, identity card, shoe”, First is packet. Second is identity card. Third is shoe. Now to reverse, change the order to: Third is shoe. Second is identity card. First is packet. So the answer is ”shoe, identity card, packet”. Joining ”shoe, identity card, packet” and ”button, driving license”, the answer is ”shoe, identity card, packet, button, driving license”.
A: ”shoe, identity card, packet, button, driving license”
QS: [EOQ]

QC: Reverse the sequence ”laptop, photo, clip”.
QS: The sequence is ”1. laptop, 2. photo, 3. clip”. The sequence is 3 items long, which is less than the minimum length of 4, so we don’t need to split it. All we need to do is reverse ”laptop, photo, clip”. First is laptop. Second is photo. Third is clip. Now to reverse, change the order to : Third is clip. Second is photo. First is laptop. So the answer is ”clip, photo, laptop”.
A: ”clip, photo, laptop”
QS: [EOQ]

QC: Reverse the sequence ”newspaper, glasses, laptop, bottle”.
QS: The sequence is ”1. newspaper, 2. glasses, 3. laptop, 4. bottle”. The sequence is 4 items long, which is equal to the minimum length of 4, so we split it. Half of 4 is 4 / 2 = 2.0. Dropping the decimal, we get that the first half will be 2 items long, ending in ”2. glasses”. The first half

(2 items) is ”1. newspaper, 2. glasses”. The first half of the sequence ends with ”2. glasses”, so the second half starts after ”2. glasses” with ”3. laptop”. The full sequence is 4 items long, and the first half is 2 items long, so the second half will be $4 - 2 = 2$ items long. The second half of the sequence (2 items) is ”3. laptop, 4. bottle”. Removing the numbers from ”1. newspaper, 2. glasses”, we get ”newspaper, glasses”. Removing the numbers from ”3. laptop, 4. bottle”, we get ”laptop, bottle”. Reversing the sequence ”newspaper, glasses”, First is newspaper. Second is glasses. Now to reverse, change the order to: Second is glasses. First is newspaper. So the answer is ”glasses, newspaper”. Reversing the sequence ”laptop, bottle”, First is laptop. Second is bottle. Now to reverse, change the order to: Second is bottle. First is laptop. So the answer is ”bottle, laptop”. Joining ”bottle, laptop” and ”glasses, newspaper”, the answer is ”bottle, laptop, glasses, newspaper”.
A: ”bottle, laptop, glasses, newspaper”
QS: [EOQ]

`reverse (tail)`

QC: Reverse the sequence ”driving license, button, packet, identity card, shoe”.
Q1: [extract] The sequence is ”1. driving license, 2. button, 3. packet, 4. identity card, 5. shoe”. The sequence is 5 items long, which is more than the minimum length of 4, so we split it. We take the first element in the sequence which is ”1. driving license”.
#1: ”1. driving license”
Q2: [extract] The full sequence is 5 items long, so the remaining sequence will be $5 - 1 = 4$ items long. The tail of the sequence with 4 items is ”2. button, 3. packet, 4. identity card, 5. shoe”.
#2: ”2. button, 3. packet, 4. identity card, 5. shoe”
Q3: [remove_numbers] Remove the numbers from #1.
#3: ”driving license”
Q4: [remove_numbers] Remove the numbers from #2.
#4: ”button, packet, identity card, shoe”
Q5: [reverse] Reverse the sequence #4.
#5: ”shoe, identity card, packet, button”
Q6: [join] #5 #3
#6: ”shoe, identity card, packet, button, driving license”
Q7: [EOQ]

QC: Reverse the sequence ”laptop, photo, clip”.
Q1: [extract] The sequence is ”1. laptop, 2. photo, 3. clip”. The sequence is 3 items long, which is less than the minimum length of 4, so we don’t need to split it. All we need to do is reverse ” laptop, photo, clip”.
#1: ”laptop, photo, clip”
Q2: [cot] Reverse the sequence #1.
#2: ”clip, photo, laptop”
Q3: [EOQ]

QC: Reverse the sequence ”newspaper, glasses, laptop, bottle”.
Q1: [extract] The sequence is ”1. newspaper, 2. glasses, 3. laptop, 4. bottle”. The sequence is 4 items long, which is more than the minimum length of 4, so we split it. We take the first element in the sequence which is ”1. newspaper”.
#1: ”1. newspaper”
Q2: [extract] The full sequence is 4 items long, so the remaining sequence will be $4 - 1 = 3$ items long. The tail of the sequence with 3 items is ”2. glasses, 3. laptop, 4. bottle”.
#2: ”2. glasses, 3. laptop, 4. bottle”
Q3: [remove_numbers] Remove the numbers from #1.
#3: ”newspaper”
Q4: [remove_numbers] Remove the numbers from #2.
#4: ”glasses, laptop, bottle”
Q5: [reverse] Reverse the sequence #4.
#5: ”bottle, laptop, glasses”
Q6: [join] #5 #3
#6: ”bottle, laptop, glasses, newspaper”
Q7: [EOQ]

G.3 LONG-DOCUMENT QA

We show one of the prompts used for CommaQA experiments here. The entire set of prompts is provided as supplemetary material.

G.3.1 DECOMPOSED PROMPTING(COARSE)

decomp

```
What awards have movies produced by people born in 1910 won?
QS: (select) [qa] Who were born in the year 1910?
A: ["Teeplemole", "Muntaril"]
QS: (project_values_flat_unique) [qa] For which movies was #1 the producer?
A: ["Featsaw", "Zalate", "Premercy"]
QS: (project_values_flat_unique) [qa] Which awards were given to #2?
A: ["Zorgion", "Chowwurst", "Hallowcock"]
QS: [EOQ]

QC: What movies have people from the country Stridery acted in?
QS: (select) [qa] Who is from the country Stridery?
A: ["Gastrat"]
QS: (project_values_flat_unique) [qa] Which movies has #1 been an actor in?
A: ["Partnershipmaker", "Nilitude", "Warpstone"]
QS: [EOQ]

QC: What awards have the actors of the Erowid winning movies received?
QS: (select) [qa] Which movies were given the Erowid award?
A: ["Dewbar", "Caudacite"]
QS: (project_values_flat_unique) [qa] Who are the actors in the movie #1?
A: ["Wetherality", "Lougerière", "Gigabut"]
QS: (project_values_flat_unique) [qa] Which awards were given to #2?
A: ["Aniconder", "Trifogation"]
QS: [EOQ]

QC: What awards did the movies directed by the Modiparity winners receive?
QS: (select) [qa] Who has been awarded the Modiparity award?
A: ["Bioperatology"]
QS: (project_values_flat_unique) [qa] Which movies has #1 directed?
A: ["Pestok", "Vitrilateral"]
QS: (project_values_flat_unique) [qa] Which awards were given to #2?
A: ["Gutney", "Antipositive"]
QS: [EOQ]

QC: What awards have movies written by people born in 1935 won?
QS: (select) [qa] Who were born in the year 1935?
A: ["Sclerocybin", "Zayage"]
QS: (project_values_flat_unique) [qa] What movies has #1 written?
A: ["Noenometer", "Tayenne", "Pneumodendron"]
QS: (project_values_flat_unique) [qa] Which awards were given to #2?
A: ["Brownbeard", "Goosehead", "Handt"]
QS: [EOQ]

QC: What movies have the directors from Legault directed?
QS: (select) [qa] Who is from the country Legault?
A: ["Metatoun", "Sapien"]
QS: (project_values_flat_unique) [qa] What movies has #1 been the director of?
A: ["Coacheship", "Misapportionment"]
QS: [EOQ]
```

qa

movie: Premercy ; directed by: Muntaril. movie: Skirtsicine ; director: Teeplemole. movie: Featsaw ; directed by: Monsterscar. movie: Zalate ; director: Monsterscar. movie: Zalate ; awarded: Hallowcock. movie: Featsaw ; awarded: Zorgion. movie: Premercy ; award: Chowwurst. movie: Skirtsicine ; award: Hallowcock. award: Goatfly ; winner: Teeplemole. person: Monsterscar ; award: Glodome. person: Muntaril ; award: Goatfly. movie: Featsaw ; release year: 1973. movie: Zalate ; release year: 1964. movie: Skirtsicine ; release year: 1973. movie: Premercy ; year: 1961. Teeplemole was an actor in the movie Skirtsicine. Muntaril was an actor in the movie Skirtsicine. Monsterscar was an actor in the movie Premercy. Muntaril was an actor in the movie Featsaw. Teeplemole was an actor in the movie Zalate. Muntaril was born in the year 1910. Teeplemole was born in 1910. Monsterscar was born in 1942. Teeplemole is from the country of Piperfish. Monsterscar is from the country of Piperfish. Muntaril is from the country of Clony. Muntaril produced the movie Skirtsicine with others. Monsterscar was one of the producers of the movie Featsaw. Monsterscar produced the movie Premercy with others. Monsterscar produced the movie Zalate with others. Teeplemole was one of the producers of the movie Featsaw. Teeplemole produced the movie Zalate with others. Muntaril produced the movie Premercy with others. Monsterscar wrote for the movie Premercy. Muntaril was one of the writers for the movie Zalate. Muntaril wrote for the movie Featsaw. Teeplemole wrote for the movie Featsaw. Monsterscar was one of the writers for the movie Zalate. Teeplemole was one of the writers for the movie Skirtsicine.
Q: For which movies was Teeplemole the producer?
A: [”Featsaw”, ”Zalate”]

Q: Which awards were given to Featsaw?
A: [”Zorgion”]

movie: Misgendery ; directed by: Wetherality. movie: Dewbar ; director: Gigabut. movie: Caudacite ; director: Lougerière. movie: Tayenne ; directed by: Lougerière. movie: Misgendery ; awarded: Microsouenesis. movie: Dewbar ; awarded: Erowid. movie: Tayenne ; awarded: Cockspit. movie: Caudacite ; award: Erowid. award: Aniconder ; winner: Wetherality. award: Aniconder ; winner: Lougerière. person: Gigabut ; award: Trifogation. movie: Dewbar ; release year: 1991. movie: Tayenne ; year: 2013. movie: Caudacite ; release year: 2008. movie: Misgendery ; year: 1991. Wetherality was an actor in the movie Dewbar. Gigabut was an actor in the movie Tayenne. Lougerière was an actor in the movie Tayenne. Lougerière acted in the movie Caudacite. Lougerière acted in the movie Misgendery. Gigabut was an actor in the movie Caudacite. Wetherality was an actor in the movie Misgendery. Wetherality was born in the year 1917. Lougerière was born in 1926. Gigabut was born in the year 1917. Gigabut grew up in the nation of Triclops. Lougerière is from the country of Tatkin. Wetherality grew up in the nation of Tatkin. Lougerière produced the movie Dewbar with others. Gigabut produced the movie Tayenne with others. Gigabut produced the movie Dewbar with others. Lougerière was one of the producers of the movie Misgendery. Wetherality was one of the producers of the movie Caudacite. Gigabut was one of the producers of the movie Caudacite. Wetherality produced the movie Misgendery with others. Wetherality produced the movie Tayenne with others. Wetherality wrote for the movie Tayenne. Gigabut wrote for the movie Misgendery. Lougerière was one of the writers for the movie Caudacite. Wetherality wrote for the movie Misgendery. Gigabut wrote for the movie Tayenne. Gigabut wrote for the movie Dewbar. Lougerière wrote for the movie Dewbar. Wetherality wrote for the movie Caudacite.
Q: Who are the actors in the movie Caudacite?
A: [”Lougerière, ”Gigabut”]

Q: Which movies were given the Erowid award?
A: [”Dewbar”, ”Caudacite”]

movie: Pastillobox ; directed by: Firmline. movie: Clenestration ; directed by: Carblock. movie: Pestok ; directed by: Bioperatology. movie: Vitrilateral ; director: Bioperatology. movie: Vitrilateral ; award: Antipositive. movie: Clenestration ; awarded: Handt. movie: Pastillobox ; awarded: Handt. movie: Pestok ; awarded: Gutney. movie: Pestok ; writer: Firmline. movie:

Clenestration ; written by: Carblock. movie: Pastillobox ; written by: Bioperatology. movie: Pestok ; writer: Bioperatology. movie: Clenestration ; written by: Firmline. movie: Vitrilateral ; writer: Bioperatology. movie: Pastillobox ; writer: Carblock. movie: Vitrilateral ; written by : Carblock. movie: Pestok ; release year: 1986. movie: Clenestration ; year: 1986. movie: Vitrilateral ; year: 1999. movie: Pastillobox ; release year: 1984. Carblock was an actor in the movie Pastillobox. Firmline was an actor in the movie Vitrilateral. Bioperatology was an actor in the movie Clenestration. Firmline acted in the movie Pastillobox. Carblock was an actor in the movie Clenestration. Bioperatology was an actor in the movie Pestok. Firmline was born in the year 1904. Bioperatology was born in the year 1935. Carblock was born in 1935. Carblock grew up in the nation of Knoppock. Firmline grew up in the nation of Tatkin. Bioperatology grew up in the nation of Tatkin. Bioperatology won the Modiparity award. Halfbill was awarded to Firmline. Halfbill was awarded to Carblock. Bioperatology was one of the producers of the movie Pestok. Bioperatology produced the movie Vitrilateral with others. Firmline produced the movie Pastillobox with others. Firmline produced the movie Clenestration with others. Carblock was one of the producers of the movie Pastillobox. Carblock produced the movie Vitrilateral with others. Carblock produced the movie Clenestration with others. Firmline was one of the producers of the movie Pestok.
Q: Who has been awarded the Modiparity award?
A: ["Bioperatology"]

Q: Which movies has Bioperatology directed?
A: ["Pestok", "Vitrilateral"]

movie: Nohit ; director: Mimicocycle. movie: Noenometer ; director: Mimicocycle. movie: Tayenne ; directed by: Zayage. movie: Pneumodendron ; director: Sclerocybin. movie: Tayenne ; awarded: Goosehead. movie: Nohit ; awarded: Handt. movie: Pneumodendron ; award: Handt. movie: Noenometer ; awarded: Brownbeard. movie: Nohit ; writer: Mimicocycle. movie: Noenometer ; written by: Sclerocybin. movie: Tayenne ; writer: Sclerocybin. movie: Pneumodendron ; written by: Zayage. movie: Tayenne ; writer: Zayage. movie: Pneumodendron ; written by: Mimicocycle. movie: Noenometer ; release year: 1991. movie: Tayenne ; year: 2013. movie: Nohit ; year: 2005. movie: Pneumodendron ; year: 2005. Mimicocycle was an actor in the movie Tayenne. Zayage acted in the movie Pneumodendron. Zayage was an actor in the movie Nohit. Sclerocybin was an actor in the movie Nohit. Sclerocybin was an actor in the movie Tayenne. Mimicocycle was an actor in the movie Noenometer. Zayage was born in 1935. Sclerocybin was born in 1935. Mimicocycle was born in 1930. Mimicocycle is from the country of Calderita. Sclerocybin grew up in the nation of Calderita. Zayage is from the country of Obility. Quinion was awarded to Zayage. Fannyxist was awarded to Sclerocybin. Fannyxist was awarded to Mimicocycle. Mimicocycle produced the movie Nohit with others. Zayage was one of the producers of the movie Nohit. Sclerocybin was one of the producers of the movie Tayenne. Sclerocybin produced the movie Pneumodendron with others. Zayage produced the movie Pneumodendron with others. Mimicocycle was one of the producers of the movie Tayenne. Sclerocybin was one of the producers of the movie Noenometer. Zayage produced the movie Noenometer with others
Q: What movies has Sclerocybin written?
A: ["Noenometer", "Tayenne"]

Q: Who were born in the year 1935?
A: ["Sclerocybin", "Zayage"]

G.3.2 DECOMPOSED PROMPTING(FINE)

`decomp`

What awards have movies produced by people born in 1910 won?
QS: (select) [simp_qa] Who were born in the year 1910?
A: ["Teeplemole", "Muntaril"]
QS: (project_values_flat_unique) [pos_qa] For which movies was #1 the producer?
A: ["Featsaw", "Zalate", "Premercy"]
QS: (project_values_flat_unique) [aw_qa] Which awards were given to #2?
A: ["Zorgion", "Chowwurst", "Hallowcock"]

QS: [EOQ]

QC: What movies have people from the country Stridery acted in?
QS: (select) [simp_qa] Who is from the country Stridery?
A: [”Gastrat”]
QS: (project_values_flat_unique) [pos_qa] Which movies has #1 been an actor in?
A: [”Partnershipmaker”, ”Nilitude”, ”Warpstone”]
QS: [EOQ]

QC: What awards have the actors of the Erowid winning movies received?
QS: (select) [aw_qa] Which movies were given the Erowid award?
A: [”Dewbar”, ”Caudacite”]
QS: (project_values_flat_unique) [pos_qa] Who are the actors in the movie #1?
A: [”Wetherality”, ”Lougerière”, ”Gigabut”]
QS: (project_values_flat_unique) [aw_qa] Which awards were given to #2?
A: [”Aniconder”, ”Trifogation”]
QS: [EOQ]

QC: What awards did the movies directed by the Modiparity winners receive?
QS: (select) [aw_qa] Who has been awarded the Modiparity award?
A: [”Bioperatology”]
QS: (project_values_flat_unique) [pos_qa] Which movies has #1 directed?
A: [”Pestok”, ”Vitrilateral”]
QS: (project_values_flat_unique) [aw_qa] Which awards were given to #2?
A: [”Gutney”, ”Antipositive”]
QS: [EOQ]

QC: What awards have movies written by people born in 1935 won?
QS: (select) [simp_qa] Who were born in the year 1935?
A: [”Sclerocybin”, ”Zayage”]
QS: (project_values_flat_unique) [pos_qa] What movies has #1 written?
A: [”Noenometer”, ”Tayenne”, ”Pneumodendron”]
QS: (project_values_flat_unique) [aw_qa] Which awards were given to #2?
A: [”Brownbeard”, ”Goosehead”, ”Handt”]
QS: [EOQ]

QC: What movies have the directors from Legault directed?
QS: (select) [simp_qa] Who is from the country Legault?
A: [”Metatoun”, ”Sapien”]
QS: (project_values_flat_unique) [pos_qa] What movies has #1 been the director of?
A: [”Coacheship”, ”Misapportionment”]
QS: [EOQ]

aw_qa

movie: Premercy ; directed by: Muntaril. movie: Skirtsicine ; director: Teeplemole. movie: Featsaw ; directed by: Monsterscar. movie: Zalate ; director: Monsterscar. movie: Zalate ; awarded: Hallowcock. movie: Featsaw ; awarded: Zorgion. movie: Premercy ; award: Chowwurst. movie: Skirtsicine ; award: Hallowcock. award: Goatfly ; winner: Teeplemole. person: Monsterscar ; award: Glodome. person: Muntaril ; award: Goatfly. movie: Featsaw ; release year: 1973. movie: Zalate ; release year: 1964. movie: Skirtsicine ; release year: 1973. movie: Premercy ; year: 1961. Teeplemole was an actor in the movie Skirtsicine. Muntaril was an actor in the movie Skirtsicine. Monsterscar was an actor in the movie Premercy. Muntaril was an actor in the movie Featsaw. Teeplemole was an actor in the movie Zalate. Muntaril was born in the year 1910. Teeplemole was born in 1910. Monsterscar was born in 1942. Teeplemole is from the country of Piperfish. Monsterscar is from the country of Piperfish. Muntaril is from the country of Clony. Muntaril produced the movie Skirtsicine with others. Monsterscar was one of the producers of the movie Featsaw. Monsterscar produced the movie Premercy with others. Monsterscar produced the movie Zalate with others. Teeplemole was one of the producers of the movie Featsaw. Teeplemole produced the movie Zalate with

others. Muntaril produced the movie Premercy with others. Monsterscar wrote for the movie Premercy. Muntaril was one of the writers for the movie Zalate. Muntaril wrote for the movie Featsaw. Teeplemole wrote for the movie Featsaw. Monsterscar was one of the writers for the movie Zalate. Teeplemole was one of the writers for the movie Skirtsicine.
Q: Which awards were given to Zalate?
A: ["Hallowcock"]

Q: Which awards were given to Featsaw?
A: ["Zorgion"]

movie: Misgendery ; directed by: Wetherality. movie: Dewbar ; director: Gigabut. movie: Caudacite ; director: Lougerière. movie: Tayenne ; directed by: Lougerière. movie: Misgendery ; awarded: Microsouenesis. movie: Dewbar ; awarded: Erowid. movie: Tayenne ; awarded: Cockspit. movie: Caudacite ; award: Erowid. award: Aniconder ; winner: Wetherality. award: Aniconder ; winner: Lougerière. person: Gigabut ; award: Trifogation. movie: Dewbar ; release year: 1991. movie: Tayenne ; year: 2013. movie: Caudacite ; release year: 2008. movie: Misgendery ; year: 1991. Wetherality was an actor in the movie Dewbar. Gigabut was an actor in the movie Tayenne. Lougerière was an actor in the movie Tayenne. Lougerière acted in the movie Caudacite. Lougerière acted in the movie Misgendery. Gigabut was an actor in the movie Caudacite. Wetherality was an actor in the movie Misgendery. Wetherality was born in the year 1917. Lougerière was born in 1926. Gigabut was born in the year 1917. Gigabut grew up in the nation of Triclops. Lougerière is from the country of Tatkin . Wetherality grew up in the nation of Tatkin. Lougerière produced the movie Dewbar with others. Gigabut produced the movie Tayenne with others. Gigabut produced the movie Dewbar with others. Lougerière was one of the producers of the movie Misgendery. Wetherality was one of the producers of the movie Caudacite. Gigabut was one of the producers of the movie Caudacite. Wetherality produced the movie Misgendery with others. Wetherality produced the movie Tayenne with others. Wetherality wrote for the movie Tayenne. Gigabut wrote for the movie Misgendery. Lougerière was one of the writers for the movie Caudacite. Wetherality wrote for the movie Misgendery. Gigabut wrote for the movie Tayenne. Gigabut wrote for the movie Dewbar. Lougerière wrote for the movie Dewbar. Wetherality wrote for the movie Caudacite.
Q: Which movies were given the Erowid award?
A: ["Dewbar", "Caudacite"]

Q: Which awards were given to Wetherality?
A: ["Aniconder"]

movie: Pastillobox ; directed by: Firmline. movie: Clenestration ; directed by: Carblock. movie: Pestok ; directed by: Bioperatology. movie: Vitrilateral ; director: Bioperatology. movie: Vitrilateral ; award: Antipositive. movie: Clenestration ; awarded: Handt. movie: Pastillobox ; awarded: Handt. movie: Pestok ; awarded: Gutney. movie: Pestok ; writer: Firmline. movie: Clenestration ; written by: Carblock. movie: Pastillobox ; written by: Bioperatology. movie: Pestok ; writer: Bioperatology. movie: Clenestration ; written by: Firmline. movie: Vitrilateral ; writer: Bioperatology. movie: Pastillobox ; writer: Carblock. movie: Vitrilateral ; written by : Carblock. movie: Pestok ; release year: 1986. movie: Clenestration ; year: 1986. movie: Vitrilateral ; year: 1999. movie: Pastillobox ; release year: 1984. Carblock was an actor in the movie Pastillobox. Firmline was an actor in the movie Vitrilateral. Bioperatology was an actor in the movie Clenestration. Firmline acted in the movie Pastillobox. Carblock was an actor in the movie Clenestration. Bioperatology was an actor in the movie Pestok. Firmline was born in the year 1904. Bioperatology was born in the year 1935. Carblock was born in 1935. Carblock grew up in the nation of Knoppock. Firmline grew up in the nation of Tatkin. Bioperatology grew up in the nation of Tatkin. Bioperatology won the Modiparity award. Halfbill was awarded to Firmline. Halfbill was awarded to Carblock. Bioperatology was one of the producers of the movie Pestok. Bioperatology produced the movie Vitrilateral with others. Firmline produced the movie Pastillobox with others. Firmline produced the movie Clenestration with others. Carblock was one of the producers of the movie Pastillobox. Carblock produced the movie Vitrilateral with others. Carblock produced the movie Clenestration with others. Firmline was one of the producers of the movie Pestok.

Q: Who has been awarded the Modiparity award?
A: [”Bioperatology”]

Q: Which awards were given to Pestok?
A: [”Gutney”]

movie: Nohit ; director: Mimicocycle. movie: Noenometer ; director: Mimicocycle. movie: Tayenne ; directed by: Zayage. movie: Pneumodendron ; director: Sclerocybin. movie: Tayenne ; awarded: Goosehead. movie: Nohit ; awarded: Handt. movie: Pneumodendron ; award: Handt. movie: Noenometer ; awarded: Brownbeard. movie: Nohit ; writer: Mimicocycle. movie: Noenometer ; written by: Sclerocybin. movie: Tayenne ; writer: Sclerocybin. movie: Pneumodendron ; written by: Zayage. movie: Tayenne ; writer: Zayage. movie: Pneumodendron ; written by: Mimicocycle. movie: Noenometer ; release year: 1991. movie: Tayenne ; year: 2013. movie: Nohit ; year: 2005. movie: Pneumodendron ; year: 2005. Mimicocycle was an actor in the movie Tayenne. Zayage acted in the movie Pneumodendron. Zayage was an actor in the movie Nohit. Sclerocybin was an actor in the movie Nohit. Sclerocybin was an actor in the movie Tayenne. Mimicocycle was an actor in the movie Noenometer. Zayage was born in 1935. Sclerocybin was born in 1935. Mimicocycle was born in 1930. Mimicocycle is from the country of Calderita. Sclerocybin grew up in the nation of Calderita. Zayage is from the country of Obility. Quinion was awarded to Zayage. Fannyxist was awarded to Sclerocybin. Fannyxist was awarded to Mimicocycle. Mimicocycle produced the movie Nohit with others. Zayage was one of the producers of the movie Nohit. Sclerocybin was one of the producers of the movie Tayenne. Sclerocybin produced the movie Pneumodendron with others. Zayage produced the movie Pneumodendron with others. Mimicocycle was one of the producers of the movie Tayenne. Sclerocybin was one of the producers of the movie Noenometer. Zayage produced the movie Noenometer with others
Q: Which awards were given to Noenometer?
A: [”Brownbeard”]

Q: Which awards were given to Pneumodendron?
A: [”Handt”]

pos_qa

movie: Premercy ; directed by: Muntaril. movie: Skirtsicine ; director: Teeplemole. movie: Featsaw ; directed by: Monsterscar. movie: Zalate ; director: Monsterscar. movie: Zalate ; awarded: Hallowcock. movie: Featsaw ; awarded: Zorgion. movie: Premercy ; award: Chowwurst. movie: Skirtsicine ; award: Hallowcock. award: Goatfly ; winner: Teeplemole. person: Monsterscar ; award: Glodome. person: Muntaril ; award: Goatfly. movie: Featsaw ; release year: 1973. movie: Zalate ; release year: 1964. movie: Skirtsicine ; release year: 1973. movie: Premercy ; year: 1961. Teeplemole was an actor in the movie Skirtsicine. Muntaril was an actor in the movie Skirtsicine. Monsterscar was an actor in the movie Premercy. Muntaril was an actor in the movie Featsaw. Teeplemole was an actor in the movie Zalate. Muntaril was born in the year 1910. Teeplemole was born in 1910. Monsterscar was born in 1942. Teeplemole is from the country of Piperfish. Monsterscar is from the country of Piperfish. Muntaril is from the country of Clony. Muntaril produced the movie Skirtsicine with others. Monsterscar was one of the producers of the movie Featsaw. Monsterscar produced the movie Premercy with others. Monsterscar produced the movie Zalate with others. Teeplemole was one of the producers of the movie Featsaw. Teeplemole produced the movie Zalate with others. Muntaril produced the movie Premercy with others. Monsterscar wrote for the movie Premercy. Muntaril was one of the writers for the movie Zalate. Muntaril wrote for the movie Featsaw. Teeplemole wrote for the movie Featsaw. Monsterscar was one of the writers for the movie Zalate. Teeplemole was one of the writers for the movie Skirtsicine.
Q: For which movies was Teeplemole the producer?
A: [”Featsaw”, ”Zalate”]

Q: For which movies was Muntaril the producer?
A: [”Premercy]

movie: Misgendery ; directed by: Wetherality. movie: Dewbar ; director: Gigabut. movie: Caudacite ; director: Lougerière. movie: Tayenne ; directed by: Lougerière. movie: Misgendery ; awarded: Microsouenesis. movie: Dewbar ; awarded: Erowid. movie: Tayenne ; awarded: Cockspit. movie: Caudacite ; award: Erowid. award: Aniconder ; winner: Wetherality. award: Aniconder ; winner: Lougerière. person: Gigabut ; award: Trifogation. movie: Dewbar ; release year: 1991. movie: Tayenne ; year: 2013. movie: Caudacite ; release year: 2008. movie: Misgendery ; year: 1991. Wetherality was an actor in the movie Dewbar. Gigabut was an actor in the movie Tayenne. Lougerière was an actor in the movie Tayenne. Lougerière acted in the movie Caudacite. Lougerière acted in the movie Misgendery. Gigabut was an actor in the movie Caudacite. Wetherality was an actor in the movie Misgendery. Wetherality was born in the year 1917. Lougerière was born in 1926. Gigabut was born in the year 1917. Gigabut grew up in the nation of Triclops. Lougerière is from the country of Tatkin . Wetherality grew up in the nation of Tatkin. Lougerière produced the movie Dewbar with others. Gigabut produced the movie Tayenne with others. Gigabut produced the movie Dewbar with others. Lougerière was one of the producers of the movie Misgendery. Wetherality was one of the producers of the movie Caudacite. Gigabut was one of the producers of the movie Caudacite. Wetherality produced the movie Misgendery with others. Wetherality produced the movie Tayenne with others. Wetherality wrote for the movie Tayenne. Gigabut wrote for the movie Misgendery. Lougerière was one of the writers for the movie Caudacite. Wetherality wrote for the movie Misgendery. Gigabut wrote for the movie Tayenne. Gigabut wrote for the movie Dewbar. Lougerière wrote for the movie Dewbar. Wetherality wrote for the movie Caudacite.
Q: Who are the actors in the movie Dewbar?
A: [”Wetherality”]

Q: Who are the actors in the movie Caudacite?
A: [”Lougerière, ”Gigabut”]

movie: Pastillobox ; directed by: Firmline. movie: Clenestration ; directed by: Carblock. movie: Pestok ; directed by: Bioperatology. movie: Vitrilateral ; director: Bioperatology. movie: Vitrilateral ; award: Antipositive. movie: Clenestration ; awarded: Handt. movie: Pastillobox ; awarded: Handt. movie: Pestok ; awarded: Gutney. movie: Pestok ; writer: Firmline. movie: Clenestration ; written by: Carblock. movie: Pastillobox ; written by: Bioperatology. movie: Pestok ; writer: Bioperatology. movie: Clenestration ; written by: Firmline. movie: Vitrilateral ; writer: Bioperatology. movie: Pastillobox ; writer: Carblock. movie: Vitrilateral ; written by : Carblock. movie: Pestok ; release year: 1986. movie: Clenestration ; year: 1986. movie: Vitrilateral ; year: 1999. movie: Pastillobox ; release year: 1984. Carblock was an actor in the movie Pastillobox. Firmline was an actor in the movie Vitrilateral. Bioperatology was an actor in the movie Clenestration. Firmline acted in the movie Pastillobox. Carblock was an actor in the movie Clenestration. Bioperatology was an actor in the movie Pestok. Firmline was born in the year 1904. Bioperatology was born in the year 1935. Carblock was born in 1935. Carblock grew up in the nation of Knoppock. Firmline grew up in the nation of Tatkin. Bioperatology grew up in the nation of Tatkin. Bioperatology won the Modiparity award. Halfbill was awarded to Firmline. Halfbill was awarded to Carblock. Bioperatology was one of the producers of the movie Pestok. Bioperatology produced the movie Vitrilateral with others. Firmline produced the movie Pastillobox with others. Firmline produced the movie Clenestration with others. Carblock was one of the producers of the movie Pastillobox. Carblock produced the movie Vitrilateral with others. Carblock produced the movie Clenestration with others. Firmline was one of the producers of the movie Pestok.
Q: Which movies has Bioperatology directed?
A: [”Pestok”, ”Vitrilateral”]

Q: Which movies has Carblock directed?
A: [”Clenestration”]

movie: Nohit ; director: Mimicocycle. movie: Noenometer ; director: Mimicocycle. movie: Tayenne ; directed by: Zayage. movie: Pneumodendron ; director: Sclerocybin. movie: Tayenne ; awarded: Goosehead. movie: Nohit ; awarded: Handt. movie: Pneumodendron ; award: Handt. movie: Noenometer ; awarded: Brownbeard. movie: Nohit ; writer:

Mimicocycle. movie: Noenometer ; written by: Sclerocybin. movie: Tayenne ; writer: Sclerocybin. movie: Pneumodendron ; written by: Zayage. movie: Tayenne ; writer: Zayage. movie: Pneumodendron ; written by: Mimicocycle. movie: Noenometer ; release year: 1991. movie: Tayenne ; year: 2013. movie: Nohit ; year: 2005. movie: Pneumodendron ; year: 2005. Mimicocycle was an actor in the movie Tayenne. Zayage acted in the movie Pneumodendron. Zayage was an actor in the movie Nohit. Sclerocybin was an actor in the movie Nohit. Sclerocybin was an actor in the movie Tayenne. Mimicocycle was an actor in the movie Noenometer. Zayage was born in 1935. Sclerocybin was born in 1935. Mimicocycle was born in 1930. Mimicocycle is from the country of Calderita. Sclerocybin grew up in the nation of Calderita. Zayage is from the country of Obility. Quinion was awarded to Zayage. Fannyxist was awarded to Sclerocybin. Fannyxist was awarded to Mimicocycle. Mimicocycle produced the movie Nohit with others. Zayage was one of the producers of the movie Nohit. Sclerocybin was one of the producers of the movie Tayenne. Sclerocybin produced the movie Pneumodendron with others. Zayage produced the movie Pneumodendron with others. Mimicocycle was one of the producers of the movie Tayenne. Sclerocybin was one of the producers of the movie Noenometer. Zayage produced the movie Noenometer with others
Q: What movies has Sclerocybin written?
A: [”Noenometer”, ”Tayenne”]

Q: What movies has Zayage written?
A: [”Pneumodendron”, ”Tayenne”]

simp_qa

movie: Premercy ; directed by: Muntaril. movie: Skirtsicine ; director: Teeplemole. movie: Featsaw ; directed by: Monsterscar. movie: Zalate ; director: Monsterscar. movie: Zalate ; awarded: Hallowcock. movie: Featsaw ; awarded: Zorgion. movie: Premercy ; award: Chowwurst. movie: Skirtsicine ; award: Hallowcock. award: Goatfly ; winner: Teeplemole. person: Monsterscar ; award: Glodome. person: Muntaril ; award: Goatfly. movie: Featsaw ; release year: 1973. movie: Zalate ; release year: 1964. movie: Skirtsicine ; release year: 1973. movie: Premercy ; year: 1961. Teeplemole was an actor in the movie Skirtsicine. Muntaril was an actor in the movie Skirtsicine. Monsterscar was an actor in the movie Premercy. Muntaril was an actor in the movie Featsaw. Teeplemole was an actor in the movie Zalate. Muntaril was born in the year 1910. Teeplemole was born in 1910. Monsterscar was born in 1942. Teeplemole is from the country of Piperfish. Monsterscar is from the country of Piperfish. Muntaril is from the country of Clony. Muntaril produced the movie Skirtsicine with others. Monsterscar was one of the producers of the movie Featsaw. Monsterscar produced the movie Premercy with others. Monsterscar produced the movie Zalate with others. Teeplemole was one of the producers of the movie Featsaw. Teeplemole produced the movie Zalate with others. Muntaril produced the movie Premercy with others. Monsterscar wrote for the movie Premercy. Muntaril was one of the writers for the movie Zalate. Muntaril wrote for the movie Featsaw. Teeplemole wrote for the movie Featsaw. Monsterscar was one of the writers for the movie Zalate. Teeplemole was one of the writers for the movie Skirtsicine.
Q: Who were born in the year 1910?
A: [”Teeplemole”, ”Muntaril”]

Q: From which country is Monsterscar?
A: [”Piperfish”]

movie: Nilitude ; director: Monsterscar. movie: Dewbar ; directed by: Metatoun. movie: Warpstone ; directed by: Gastrat. movie: Partnershipmaker ; director: Metatoun. movie: Dewbar ; award: Tachychronograph. movie: Partnershipmaker ; awarded: Tachychronograph. movie: Nilitude ; award: Paleodactyl. movie: Warpstone ; award: Sabonade. person: Gastrat ; award: Trifogation. award: Polyquadrase ; winner: Monsterscar. award: Trifogation ; winner: Metatoun. movie: Warpstone ; release year: 1956. movie: Dewbar ; release year: 1984. movie: Nilitude ; year: 1984. movie: Partnershipmaker ; year: 1962. Gastrat was an actor in the movie Partnershipmaker. Metatoun was an actor in the movie Partnershipmaker. Metatoun was an actor in the movie Nilitude. Gastrat acted in the movie Nilitude. Monsterscar was an actor in the movie Dewbar. Gastrat acted in the movie Warpstone. Metatoun acted in the movie Warpstone. Metatoun was born in 1939. Gastrat was born in the year 1933.

Monsterscar was born in 1933. Metatoun grew up in the nation of Moulole. Gastrat is from the country of Stridery. Monsterscar grew up in the nation of Moulole. Monsterscar produced the movie Nilitude with others. Monsterscar was one of the producers of the movie Warpstone . Metatoun was one of the producers of the movie Warpstone. Gastrat was one of the producers of the movie Nilitude. Metatoun produced the movie Partnershipmaker with others. Metatoun produced the movie Dewbar with others. Monsterscar was one of the producers of the movie Partnershipmaker. Gastrat produced the movie Dewbar with others. Metatoun wrote for the movie Partnershipmaker. Gastrat wrote for the movie Warpstone. Gastrat was one of the writers for the movie Dewbar. Monsterscar was one of the writers for the movie Nilitude. Metatoun wrote for the movie Warpstone.
Q: Who is from the country Stridery?
A: [”Gastrat”]

Q: Which movies were released in 1984?
A: [”Dewbar”, ”Nilitude”]

movie: Nohit ; director: Mimicocycle. movie: Noenometer ; director: Mimicocycle. movie: Tayenne ; directed by: Zayage. movie: Pneumodendron ; director: Sclerocybin. movie: Tayenne ; awarded: Goosehead. movie: Nohit ; awarded: Handt. movie: Pneumodendron ; award: Handt. movie: Noenometer ; awarded: Brownbeard. movie: Nohit ; writer: Mimicocycle. movie: Noenometer ; written by: Sclerocybin. movie: Tayenne ; writer: Sclerocybin. movie: Pneumodendron ; written by: Zayage. movie: Tayenne ; writer: Zayage. movie: Pneumodendron ; written by: Mimicocycle. movie: Noenometer ; release year: 1991. movie: Tayenne ; year: 2013. movie: Nohit ; year: 2005. movie: Pneumodendron ; year: 2005. Mimicocycle was an actor in the movie Tayenne. Zayage acted in the movie Pneumodendron. Zayage was an actor in the movie Nohit. Sclerocybin was an actor in the movie Nohit. Sclerocybin was an actor in the movie Tayenne. Mimicocycle was an actor in the movie Noenometer. Zayage was born in 1935. Sclerocybin was born in 1935. Mimicocycle was born in 1930. Mimicocycle is from the country of Calderita. Sclerocybin grew up in the nation of Calderita. Zayage is from the country of Obility. Quinion was awarded to Zayage. Fannyxist was awarded to Sclerocybin. Fannyxist was awarded to Mimicocycle. Mimicocycle produced the movie Nohit with others. Zayage was one of the producers of the movie Nohit. Sclerocybin was one of the producers of the movie Tayenne. Sclerocybin produced the movie Pneumodendron with others. Zayage produced the movie Pneumodendron with others. Mimicocycle was one of the producers of the movie Tayenne. Sclerocybin was one of the producers of the movie Noenometer. Zayage produced the movie Noenometer with others
Q: Who were born in the year 1935?
A: [”Sclerocybin”, ”Zayage”]

Q: When was the movie Nohit released?
A: [”2005”]

movie: Coacheship ; director: Metatoun. movie: Assamplifier ; director: Kapod. movie: Misapportionment ; director: Sapien. movie: Quinsid ; director: Kapod. movie: Assamplifier ; award: Zorgion. movie: Quinsid ; awarded: Airpipe. movie: Coacheship ; award: Electrodesal. movie: Misapportionment ; award: Airpipe. movie: Coacheship ; written by: Metatoun. movie: Misapportionment ; written by: Kapod. movie: Coacheship ; written by: Kapod. movie : Quinsid ; writer: Sapien. movie: Misapportionment ; written by: Metatoun. movie: Assamplifier ; written by: Kapod. movie: Assamplifier ; written by: Sapien. movie: Assamplifier ; release year: 2000. movie: Coacheship ; year: 2001. movie: Quinsid ; year: 2005. movie: Misapportionment ; year: 2005. Sapien was an actor in the movie Misapportionment. Sapien acted in the movie Assamplifier. Kapod acted in the movie Quinsid . Sapien acted in the movie Coacheship. Metatoun was an actor in the movie Quinsid. Kapod acted in the movie Misapportionment. Metatoun acted in the movie Coacheship. Kapod acted in the movie Assamplifier. Sapien was born in the year 1910. Metatoun was born in 1928. Kapod was born in the year 1910. Metatoun is from the country of Legault. Kapod grew up in the nation of Tatkin. Sapien is from the country of Legault. Malwarp was awarded to Sapien. Metatoun won the Monkeynote award. Kapod won the Monkeynote award. Kapod was one of the producers of the movie Quinsid. Metatoun was one of the producers of the movie

Misapportionment. Metatoun was one of the producers of the movie Quinsid. Sapien was one of the producers of the movie Assamplifier. Sapien produced the movie Coacheship with others. Metatoun was one of the producers of the movie Assamplifier. Kapod was one of the producers of the movie Misapportionment. Kapod was one of the producers of the movie Coacheship.
Q: Who is from the country Legault?
A: [”Metatoun”, ”Sapien”]

Q: When was Kapod born?
A: [”1910”]

G.3.3 COT

COT

movie: Premercy ; directed by: Muntaril. movie: Skirtsicine ; director: Teeplemole. movie: Featsaw ; directed by: Monsterscar. movie: Zalate ; director: Monsterscar. movie: Zalate ; awarded: Hallowcock. movie: Featsaw ; awarded: Zorgion. movie: Premercy ; award: Chowwurst. movie: Skirtsicine ; award: Hallowcock. award: Goatfly ; winner: Teeplemole. person: Monsterscar ; award: Glodome. person: Muntaril ; award: Goatfly. movie: Featsaw ; release year: 1973. movie: Zalate ; release year: 1964. movie: Skirtsicine ; release year: 1973. movie: Premercy ; year: 1961. Teeplemole was an actor in the movie Skirtsicine. Muntaril was an actor in the movie Skirtsicine. Monsterscar was an actor in the movie Premercy. Muntaril was an actor in the movie Featsaw. Teeplemole was an actor in the movie Zalate. Muntaril was born in the year 1910. Teeplemole was born in 1910. Monsterscar was born in 1942. Teeplemole is from the country of Piperfish. Monsterscar is from the country of Piperfish. Muntaril is from the country of Clony. Muntaril produced the movie Skirtsicine with others. Monsterscar was one of the producers of the movie Featsaw. Monsterscar produced the movie Premercy with others. Monsterscar produced the movie Zalate with others. Teeplemole was one of the producers of the movie Featsaw. Teeplemole produced the movie Zalate with others. Muntaril produced the movie Premercy with others. Monsterscar wrote for the movie Premercy. Muntaril was one of the writers for the movie Zalate. Muntaril wrote for the movie Featsaw. Teeplemole wrote for the movie Featsaw. Monsterscar was one of the writers for the movie Zalate. Teeplemole was one of the writers for the movie Skirtsicine.
QC: What awards have movies produced by people born in 1910 won?
QS: The people born in 1910 were Muntaril and Teeplemole. Teeplemole produced the movies Featsaw and Zalate. Muntaril produced the movie Premercy. Featsaw was awarded the Zorgion award. Premercy was awarded the Chowwurst award. Zalate was awarded the Hallowcock award. So the answer is [”Zorgion”, ”Chowwurst”, ”Hallowcock”].
A: [”Zorgion”, ”Chowwurst”, ”Hallowcock”]
QS: [EOQ]

movie: Misgendery ; directed by: Wetherality. movie: Dewbar ; director: Gigabut. movie: Caudacite ; director: Lougerière. movie: Tayenne ; directed by: Lougerière. movie: Misgendery ; awarded: Microsouenesis. movie: Dewbar ; awarded: Erowid. movie: Tayenne ; awarded: Cockspit. movie: Caudacite ; award: Erowid. award: Aniconder ; winner: Wetherality. award: Aniconder ; winner: Lougerière. person: Gigabut ; award: Trifogation. movie: Dewbar ; release year: 1991. movie: Tayenne ; year: 2013. movie: Caudacite ; release year: 2008. movie: Misgendery ; year: 1991. Wetherality was an actor in the movie Dewbar. Gigabut was an actor in the movie Tayenne. Lougerière was an actor in the movie Tayenne. Lougerière acted in the movie Caudacite. Lougerière acted in the movie Misgendery. Gigabut was an actor in the movie Caudacite. Wetherality was an actor in the movie Misgendery. Wetherality was born in the year 1917. Lougerière was born in 1926. Gigabut was born in the year 1917. Gigabut grew up in the nation of Triclops. Lougerière is from the country of Tatkin. Wetherality grew up in the nation of Tatkin. Lougerière produced the movie Dewbar with others. Gigabut produced the movie Tayenne with others. Gigabut produced the movie Dewbar with others. Lougerière was one of the producers of the movie Misgendery. Wetherality was one of the producers of the movie Caudacite. Gigabut was one of the producers of the movie Caudacite. Wetherality produced the movie Misgendery with others. Wetherality produced the movie Tayenne with others. Wetherality wrote for the movie

Tayenne. Gigabut wrote for the movie Misgendery. Lougerière was one of the writers for the movie Caudacite. Wetherality wrote for the movie Misgendery. Gigabut wrote for the movie Tayenne. Gigabut wrote for the movie Dewbar. Lougerière wrote for the movie Dewbar. Wetherality wrote for the movie Caudacite.
QC: What awards have the actors of the Erowid winning movies received?
QS: The movies that won the Erowid award are Dewbar and Caudacite. Wetherality was an actor in Dewbar. Lougerière and Gigabut acted in Caudacite. Wetherality won the Aniconder award. Lougerière also won the Aniconder award. Gigabut won the Trifogation award. So the answer is [”Aniconder”, ”Trifogation”].
A: [”Aniconder”, ”Trifogation”]
QS: [EOQ]

movie: Pastillobox ; directed by: Firmline. movie: Clenestration ; directed by: Carblock. movie: Pestok ; directed by: Bioperatology. movie: Vitrilateral ; director: Bioperatology. movie: Vitrilateral ; award: Antipositive. movie: Clenestration ; awarded: Handt. movie: Pastillobox ; awarded: Handt. movie: Pestok ; awarded: Gutney. movie: Pestok ; writer: Firmline. movie: Clenestration ; written by: Carblock. movie: Pastillobox ; written by: Bioperatology. movie: Pestok ; writer: Bioperatology. movie: Clenestration ; written by: Firmline. movie: Vitrilateral ; writer: Bioperatology. movie: Pastillobox ; writer: Carblock. movie: Vitrilateral ; written by : Carblock. movie: Pestok ; release year: 1986. movie: Clenestration ; year: 1986. movie: Vitrilateral ; year: 1999. movie: Pastillobox ; release year: 1984. Carblock was an actor in the movie Pastillobox. Firmline was an actor in the movie Vitrilateral. Bioperatology was an actor in the movie Clenestration. Firmline acted in the movie Pastillobox. Carblock was an actor in the movie Clenestration. Bioperatology was an actor in the movie Pestok. Firmline was born in the year 1904. Bioperatology was born in the year 1935. Carblock was born in 1935. Carblock grew up in the nation of Knoppock. Firmline grew up in the nation of Tatkin. Bioperatology grew up in the nation of Tatkin. Bioperatology won the Modiparity award. Halfbill was awarded to Firmline. Halfbill was awarded to Carblock. Bioperatology was one of the producers of the movie Pestok. Bioperatology produced the movie Vitrilateral with others. Firmline produced the movie Pastillobox with others. Firmline produced the movie Clenestration with others. Carblock was one of the producers of the movie Pastillobox. Carblock produced the movie Vitrilateral with others. Carblock produced the movie Clenestration with others. Firmline was one of the producers of the movie Pestok.
QC: What awards did the movies directed by the Modiparity winners receive?
QS: The Modiparity winners are Bioperatology. Bioperatology has directed the movies Pestok and Vitrilateral. Pestok has won the Gutney award. Vitrilateral has won the Antipositive award. So the answer is [”Gutney”, ”Antipositive”].
A: [”Gutney”, ”Antipositive”]
QS: [EOQ]

movie: Nohit ; director: Mimicocycle. movie: Noenometer ; director: Mimicocycle. movie: Tayenne ; directed by: Zayage. movie: Pneumodendron ; director: Sclerocybin. movie: Tayenne ; awarded: Goosehead. movie: Nohit ; awarded: Handt. movie: Pneumodendron ; award: Handt. movie: Noenometer ; awarded: Brownbeard. movie: Nohit ; writer: Mimicocycle. movie: Noenometer ; written by: Sclerocybin. movie: Tayenne ; writer: Sclerocybin. movie: Pneumodendron ; written by: Zayage. movie: Tayenne ; writer: Zayage. movie: Pneumodendron ; written by: Mimicocycle. movie: Noenometer ; release year: 1991. movie: Tayenne ; year: 2013. movie: Nohit ; year: 2005. movie: Pneumodendron ; year: 2005. Mimicocycle was an actor in the movie Tayenne. Zayage acted in the movie Pneumodendron. Zayage was an actor in the movie Nohit. Sclerocybin was an actor in the movie Nohit. Sclerocybin was an actor in the movie Tayenne. Mimicocycle was an actor in the movie Noenometer. Zayage was born in 1935. Sclerocybin was born in 1935. Mimicocycle was born in 1930. Mimicocycle is from the country of Calderita. Sclerocybin grew up in the nation of Calderita. Zayage is from the country of Obility. Quinion was awarded to Zayage. Fannyxist was awarded to Sclerocybin. Fannyxist was awarded to Mimicocycle. Mimicocycle produced the movie Nohit with others. Zayage was one of the producers of the movie Nohit. Sclerocybin was one of the producers of the movie Tayenne. Sclerocybin produced the movie Pneumodendron with others. Zayage produced the movie Pneumodendron with others.

Mimicocycle was one of the producers of the movie Tayenne. Sclerocybin was one of the producers of the movie Noenometer. Zayage produced the movie Noenometer with others
QC: What awards have movies written by people born in 1935 won?
QS: The people born in 1935 were Sclerocybin and Zayage. Sclerocybin has written the movies Noenometer and Tayenne. Zayage has written the movies Pneumodendron and Tayenne. Noenometer has won the Brownbeard award. Tayenne has won the Goosehead award. Pneumodendron has won the Handt award. So the answer is [”Brownbeard”, ”Goosehead”, ”Handt”].
A: [”Brownbeard”, ”Goosehead”, ”Handt”]
QS: [EOQ]

G.4 MATH QA

cot

Q: There are 15 trees in the grove. Grove workers will plant trees in the grove today. After they are done, there will be 21 trees. How many trees did the grove workers plant today?
A: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 − 15 = 6. The answer is 6.

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot ?
A: There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5. The answer is 5.

Q: Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?
A: Originally, Leah had 32 chocolates. Her sister had 42. So in total they had 32 + 42 = 74. After eating 35, they had 74 − 35 = 39. The answer is 39.

Q: Jason had 20 lollipops. He gave Denny some lollipops. Now Jason has 12 lollipops. How many lollipops did Jason give to Denny?
A: Jason started with 20 lollipops. Then he had 12 after giving some to Denny. So he gave Denny 20 − 12 = 8. The answer is 8.

Q: Shawn has five toys. For Christmas, he got two toys each from his mom and dad. How many toys does he have now?
A: Shawn started with 5 toys. If he got 2 toys each from his mom and dad, then that is 4 more toys. 5 + 4 = 9. The answer is 9.

Q: There were nine computers in the server room. Five more computers were installed each day, from monday to thursday. How many computers are now in the server room?
A: There were originally 9 computers. For each of 4 days, 5 more computers were added. So 5 $*$ 4 = 20 computers were added. 9 + 20 is 29. The answer is 29.

Q: Michael had 58 golf balls. On tuesday, he lost 23 golf balls. On wednesday, he lost 2 more. How many golf balls did he have at the end of wednesday?
A: Michael started with 58 golf balls. After losing 23 on tuesday, he had 58 − 23 = 35. After losing 2 more, he had 35 − 2 = 33 golf balls. The answer is 33.

Q: Olivia has $23. She bought five bagels for $3 each. How much money does she have left?
A: Olivia had 23 dollars. 5 bagels for 3 dollars each will be 5 x 3 = 15 dollars. So she has 23 − 15 dollars left. 23 − 15 is 8. The answer is 8.

gpt_ans

Q: There are 15 trees originally. Then there were 21 trees after some more were planted. So there must have been 21 − 15 = 6 trees planted.
A: 6

Q: There are originally 3 cars. 2 more cars arrive. 3 + 2 = 5. The answer is 5.

A: 5

Q: Originally, Leah had 32 chocolates. Her sister had 42. So in total they had 32 + 42 = 74. After eating 35, they had 74 − 35 = 39. The answer is 39.
A: 39

Q: Jason started with 20 lollipops. Then he had 12 after giving some to Denny. So he gave Denny 20 − 12 = 8 apples.
A: 8

Q: Shawn started with 5 toys. If he got 2 toys each from his mom and dad, then that is 4 more toys. 5 + 4 = 9. The answer is 9.
A: 9

Q: There were originally 9 computers. For each of 4 days, 5 more computers were added. So 5 ∗ 4 = 20 computers were added. 9 + 20 is 29. The answer is 29.
A: 29

Q: Michael started with 58 golf balls. After losing 23 on tuesday, he had 58 − 23 = 35. After losing 2 more, he had 35 − 2 = 33 golf balls.
A: 33

Q: Olivia had 23 dollars. 5 bagels for 3 dollars each will be 5 x 3 = 15 dollars. So she has 23 − 15 = 8 dollars left.
A: 8

G.5 HOTPOTQA

The prompts in this section implement Decomposed Prompting approach to open-domain multihop QA. For brevity we've included prompts for 5 of 25 randomly sampled questions. The full prompts are attached with the submission and will also be released with the code. Note that we selected a set of 100 questions from the development set to tune the hyperparameter (number of paragraphs to retrieve for all of the retrieval-based approaches).

decomp

QC: who is older Jeremy Horn or Renato Sobral ?
QS: (select) [retrieve_odqa] When was Jeremy Horn born?
A: {"titles": ["Jeremy Horn", "Train horn"], ["August 25, 1975"]}
QS: (select) [retrieve_odqa] When was Renato Sobral born?
A: {"titles": ["Sobral Base", Renato Sobral", "Luta Livre"], ["September 7, 1975"]}
QS: (select) [multihop_titleqa] Titles: ["Jeremy Horn", "Train horn", "Sobral Base", Renato Sobral ", "Luta Livre"]. Question: Who is older Jeremy Horn or Renato Sobral?
A: ["Jeremy Horn"]
QS: [EOQ]

QC: In which country did this Australian who was detained in Guantanamo Bay detention camp and published "Guantanamo: My Journey" receive para−military training?
QS: (select) [retrieve_odqa] Which Australian was detained in Guantanamo Bay detention camp and published "Guantanamo: My Journey"?
A: {"titles": ["Shawali Khan", "Guantanamo: My Journey", "Yasim Muhammed Basardah"], " answer": ["David Hicks"]}
QS: (select) [retrieve_odqa] In which country did David Hicks receive para−military training?
A: {"titles": ["David Hicks", "Train (military)"], "answer": ["Afghanistan"]}
QS: (select) [multihop_titleqa] Titles: ["Shawali Khan", "Guantanamo: My Journey", "Yasim Muhammed Basardah", "David Hicks", "Train (military)"]. Question: In which country did this Australian who was detained in Guantanamo Bay detention camp and published " Guantanamo: My Journey" receive para−military training?
A: ["Afghanistan"]
QS: [EOQ]

QC: Which former Canadian Professional Ice Hockey Player endorses MLX Skates?
QS: (select) [retrieve_odqa] Who endorses MLX Skates?
A: {"titles": ["Marit Henie", "MLX Skates", "Puma Clyde", "MLX (gene)"], ["Mario Lemieux"]}
QS: (select) [retrieve_odqa] Is Mario Lemieux a former Canadian Professional Ice Hockey Player?
A: {"titles": ["Mario Lemieux", "Claude Lemieux", "Mario Simioni"], ["yes"]}
QS: (select) [multihop_titleqa] Titles: ["Marit Henie", "MLX Skates", "Puma Clyde", "MLX (gene)", "Mario Lemieux", "Claude Lemieux", "Mario Simioni"]. Question: Which former Canadian Professional Ice Hockey Player endorses MLX Skates?
A: ["Mario Lemieux"]

QC: Mister Magoo's Christmas Carol was produced by the same studio that produced a film that featured the only animated−film role by who?
QS: (select) [retrieve_odqa] Mister Magoo's Christmas Carol was produced by what studio?
A: {"titles": ["UPA (animation studio)", "ImageMovers", "Jane Kean", "List of Christmas carols "], ["United Productions of America"]}
QS: (select) [retrieve_odqa] United Productions of America studio produced a film that featured the only animated−film role by who?
A: {"titles": ["Gay Purr−ee", "Walt Disney Pictures", "Action Synthese", "A. Film A/S"], ["Judy Garland"]}
QS: (select) [multihop_titleqa] Titles: ["UPA (animation studio)", "ImageMovers", "Jane Kean", " List of Christmas carols", "Gay Purr−ee", "Walt Disney Pictures", "Action Synthese", "A. Film A/S"]. Question: Mister Magoo's Christmas Carol was produced by the same studio that produced a film that featured the only animated−film role by who?
A: ["Judy Garland"]
QS: [EOQ]

QC: What was the 2014 population of the city where Lake Wales Medical Center is located?
QS: (select) [retrieve_odqa] In which city is Lake Wales Medical Center is located?
A: {"titles": ["Lake Wales Medical Center", "Salt Lake Regional Medical Center", "Sky Lakes Medical Center", "Which Medical Device", "Lake Wales Medical Center"], "answer": ["Lake Wales"]}
QS: (select) [retrieve_odqa] What was the population of Lake Wales in 2014?
A: {"titles": ["Lake Wales, Florida", "Wales Act 2014", "2014 in Wales", "Lake Albert (New South Wales)", "Saddlebag Lake Resort", "List of localities in Wales by population"], " answer": ["15,140"]}
QS: (select) [multihop_titleqa] Titles: ["Lake Wales Medical Center", "Salt Lake Regional Medical Center", "Sky Lakes Medical Center", "Which Medical Device", "Lake Wales Medical Center", "Lake Wales, Florida", "Wales Act 2014", "2014 in Wales", "Lake Albert (New South Wales)", "Saddlebag Lake Resort", "List of localities in Wales by population"]. Question: What was the 2014 population of the city where Lake Wales Medical Center is located?
A: ["15,140"]
QS: [EOQ]

retrieve_odqa

QC: When was Jeremy Horn born?
QS: (select) [retrieve] When was Jeremy Horn born?
A: ["Jeremy Horn", "Train horn"]
QS: (select) [singlehop_titleqa] Titles: ["Jeremy Horn", "Train horn"]. Question: When was Jeremy Horn born?
A: {"titles": ["Jeremy Horn", "Train horn"], ["August 25, 1975"]}
QS: [EOQ]

QC: When was Renato Sobral born?
QS: (select) [retrieve] When was Renato Sobral born?
A: ["Sobral Base", Renato Sobral", "Luta Livre"]
QS: (select) [singlehop_titleqa] Titles: ["Sobral Base", Renato Sobral", "Luta Livre"]. Question: When was Renato Sobral born?

A: {”titles”: [”Sobral Base”, Renato Sobral”, ”Luta Livre”], [”September 7, 1975”]}
QS: [EOQ]

QC: In which country did David Hicks receive para−military training?
QS: (select) [retrieve] In which country did David Hicks receive para−military training?
A: [”David Hicks”, ”Train (military)”]
QS: (select) [singlehop_titleqa] Titles: [”David Hicks”, ”Train (military)”]. Question: In which country did David Hicks receive para−military training?
A: {”titles”: [”David Hicks”, ”Train (military)”], ”answer”: [”Afghanistan”]}
QS: [EOQ]

QC: Which Australian was detained in Guantanamo Bay detention camp and published ” Guantanamo: My Journey”?
QS: (select) [retrieve] Which Australian was detained in Guantanamo Bay detention camp and published ”Guantanamo: My Journey”?
A: [”Shawali Khan”, ”Guantanamo: My Journey”, ”Yasim Muhammed Basardah”]
QS: (select) [singlehop_titleqa] Titles: [”Shawali Khan”, ”Guantanamo: My Journey”, ”Yasim Muhammed Basardah”]. Question: Which Australian was detaindin Guantanamo Bay detention camp and published ”Guantanamo: My Journey”?
A: {”titles”: [”Shawali Khan”, ”Guantanamo: My Journey”, ”Yasim Muhammed Basardah”], ” answer”: [”David Hicks”]}
QS: [EOQ]

QC: Who endorses MLX Skates?
QS: (select) [retrieve] Who endorses MLX Skates?
A: [”Marit Henie”, ”MLX Skates”, ”Puma Clyde”, ”MLX (gene)”]
QS: (select) [singlehop_titleqa] Titles: [”Marit Henie”, ”MLX Skates”, ”Puma Clyde”, ”MLX (gene)”]. Question: Who endorses MLX Skates?
A: {”titles”: [”Marit Henie”, ”MLX Skates”, ”Puma Clyde”, ”MLX (gene)”], [”Mario Lemieux”]}
QS: [EOQ]

QC: Is Mario Lemieux a former Canadian Professional Ice Hockey Player?
QS: (select) [retrieve] Is Mario Lemieux a former Canadian Professional Ice Hockey Player?
A: [”Mario Lemieux”, ”Claude Lemieux”, ”Mario Simioni”]
QS: (select) [singlehop_titleqa] Titles: [”Mario Lemieux”, ”Claude Lemieux”, ”Mario Simioni”]. Question: Is Mario Lemieux a former Canadian Professional Ice Hockey Player?
A: {”titles”: [”Mario Lemieux”, ”Claude Lemieux”, ”Mario Simioni”], [”yes”]}
QS: [EOQ]

QC: Mister Magoo’s Christmas Carol was produced by what studio?
QS: (select) [retrieve] Mister Magoo’s Christmas Carol was produced by what studio?
A: [”UPA (animation studio)”, ”ImageMovers”, ”Jane Kean”, ”List of Christmas carols”]
QS: (select) [singlehop_titleqa] Titles: [”UPA (animation studio)”, ”ImageMovers”, ”Jane Kean”, ” List of Christmas carols”]. Question: Mister Magoo’s Christmas Carol was produced by what studio?
A: {”titles”: [”UPA (animation studio)”, ”ImageMovers”, ”Jane Kean”, ”List of Christmas carols ”], [”United Productions of America”]}
QS: [EOQ]

QC: United Productions of America studio produced a film that featured the only animated−film role by who?
QS: (select) [retrieve] United Productions of America studio produced a film that featured the only animated−film role by who?
A: [”Gay Purr−ee”, ”Walt Disney Pictures”, ”Action Synthese”, ”A. Film A/S”]
QS: (select) [singlehop_titleqa] Titles: [”Gay Purr−ee”, ”Walt Disney Pictures”, ”Action Synthese ”, ”A. Film A/S”]. Question: United Productions of America studio produced a film that featured the only animated−film role by who?
A: {”titles”: [”Gay Purr−ee”, ”Walt Disney Pictures”, ”Action Synthese”, ”A. Film A/S”], [”Judy Garland”]}

QS: [EOQ]

QC: What was the population of Lake Wales in 2014?
QS: (select) [retrieve] What was the population of Lake Wales in 2014?
A: [”Lake Wales, Florida”, ”Wales Act 2014”, ”2014 in Wales”, ”Lake Albert (New South Wales)”, ”Saddlebag Lake Resort”, ”List of localities in Wales by population”]
QS: (select) [singlehop_titleqa] Titles: [”Lake Wales, Florida”, ”Wales Act 2014”, ”2014 in Wales”, ”Lake Albert (New South Wales)”, ”Saddlebag Lake Resort”, ”List of localities in Wales by population”]. Question: What was the population of Lake Wales in 2014?
A: {”titles”: [”Lake Wales, Florida”, ”Wales Act 2014”, ”2014 in Wales”, ”Lake Albert (New South Wales)”, ”Saddlebag Lake Resort”, ”List of localities in Wales by population”], ”answer”: [”15,140”]}
QS: [EOQ]

QC: In which city is Lake Wales Medical Center is located?
QS: (select) [retrieve] In which city is Lake Wales Medical Center is located?
A: [”Lake Wales Medical Center”, ”Salt Lake Regional Medical Center”, ”Sky Lakes Medical Center”, ”Which Medical Device”, ”Lake Wales Medical Center”]
QS: (select) [singlehop_titleqa] Titles: [”Lake Wales Medical Center”, ”Salt Lake Regional Medical Center”, ”Sky Lakes Medical Center”, ”Which Medical Device”, ”Lake Wales Medical Center”]. Question: In which city is Lake Wales Medical Center is located?
A: {”titles”: [”Lake Wales Medical Center”, ”Salt Lake Regional Medical Center”, ”Sky Lakes Medical Center”, ”Which Medical Device”, ”Lake Wales Medical Center”], ”answer”: [”Lake Wales”]}
QS: [EOQ]

`singlehop_titleqa`

Wikipedia Title: Jeremy Horn
<hidden for brevity>

Wikipedia Title: Renato Sobral
<hidden for brevity>

Wikipedia Title: Renato Aragao
<hidden for brevity>

Q: When was Jeremy Horn born?
A: [”August 25, 1975”]

Q: When was Renato Sobral born?
A: [”September 7, 1975”]

Wikipedia Title: Guantanamo: My Journey
<hidden for brevity>

Wikipedia Title: David Hicks
<hidden for brevity>

Wikipedia Title: John Adams Project
<hidden for brevity>

Q: Which Australian was detained in Guantanamo Bay detention camp and published ”Guantanamo: My Journey”?
A: [”David Hicks”]

Q: In which country did David Hicks receive para−military training?
A: [”Afghanistan”]

Wikipedia Title: Brian Smith (ice hockey, born 1940)

<hidden for brevity>

Wikipedia Title: MLX Skates
<hidden for brevity>

Wikipedia Title: Mario Lemieux
<hidden for brevity>

Q: Is Mario Lemieux a former Canadian Professional Ice Hockey Player?
A: ["yes"]

Q: Who endorses MLX Skates?
A: ["Mario Lemieux"]

Wikipedia Title: Gay Purr−ee
<hidden for brevity>

Wikipedia Title: A Christmas Carol
<hidden for brevity>

Wikipedia Title: UPA (animation studio)
<hidden for brevity>

Q: Mister Magoo's Christmas Carol was produced by what studio?
A: ["United Productions of America"]

Q: United Productions of America studio produced a film that featured the only animated−film role by who?
A: ["Judy Garland"]

Wikipedia Title: Lake Wales Medical Center
<hidden for brevity>

Wikipedia Title: Church of the Holy Spirit (Lake Wales, Florida)
<hidden for brevity>

Wikipedia Title: Lake Wales, Florida
<hidden for brevity>

Q: In which city is Lake Wales Medical Center is located?
A: ["Lake Wales"]

Q: What was the population of Lake Wales in 2014?
A: ["15,140"]

`multihop_titleqa (direct)`

Wikipedia Title: Jeremy Horn
<hidden for brevity>

Wikipedia Title: Renato Sobral
<hidden for brevity>

Wikipedia Title: Renato Aragao
<hidden for brevity>

Q: who is older Jeremy Horn or Renato Sobral ?
A: ["Jeremy Horn"]

Wikipedia Title: Guantanamo: My Journey

<hidden for brevity>

Wikipedia Title: David Hicks
<hidden for brevity>

Wikipedia Title: John Adams Project
<hidden for brevity>

Q: In which country did this Australian who was detained in Guantanamo Bay detention camp and published ”Guantanamo: My Journey” receive para−military training?
A: [”Afghanistan”]

Wikipedia Title: Brian Smith (ice hockey, born 1940)
<hidden for brevity>

Wikipedia Title: MLX Skates
<hidden for brevity>

Wikipedia Title: Mario Lemieux
<hidden for brevity>

Q: Which former Canadian Professional Ice Hockey Player endorses MLX Skates?
A: [”Mario Lemieux”]

Wikipedia Title: Gay Purr−ee
<hidden for brevity>

Wikipedia Title: A Christmas Carol
<hidden for brevity>

Wikipedia Title: UPA (animation studio)
<hidden for brevity>

Q: Mister Magoo’s Christmas Carol was produced by the same studio that produced a film that featured the only animated−film role by who?
A: [”Judy Garland”]

Wikipedia Title: Lake Wales Medical Center
<hidden for brevity>

Wikipedia Title: Church of the Holy Spirit (Lake Wales, Florida)
<hidden for brevity>

Wikipedia Title: Lake Wales, Florida
<hidden for brevity>

Q: What was the 2014 population of the city where Lake Wales Medical Center is located?
A: [”15,140”]

`multihop_titleqa (cot)`

Wikipedia Title: Jeremy Horn
<hidden for brevity>

Wikipedia Title: Renato Sobral
<hidden for brevity>

Wikipedia Title: Renato Aragao
<hidden for brevity>

Q: who is older Jeremy Horn or Renato Sobral ?
A: Jeremy Horn was born on August 25, 1975. Renato Sobral was born on September 7, 1975. Thus, Jeremy Horn is older. So the answer is: Jeremy Horn.

Wikipedia Title: Guantanamo: My Journey
<hidden for brevity>

Wikipedia Title: David Hicks
<hidden for brevity>

Wikipedia Title: John Adams Project
<hidden for brevity>

Q: In which country did this Australian who was detained in Guantanamo Bay detention camp and published "Guantanamo: My Journey" receive para−military training?
A: The Australian who was detained in Guantanamo Bay detention camp and published " Guantanamo: My Journey" is David Hicks. David Hicks received his para−military training in Afghanistan. So the answer is: Afghanistan.

Wikipedia Title: Brian Smith (ice hockey, born 1940)
<hidden for brevity>

Wikipedia Title: MLX Skates
<hidden for brevity>

Wikipedia Title: Mario Lemieux
<hidden for brevity>

Q: Which former Canadian Professional Ice Hockey Player endorses MLX Skates?
A: MLX Skates is endorsed by Mario Lemieux. Mario Lemieux is a former Canadian Professional Ice Hockey Player. So the answer is: Mario Lemieux.

Wikipedia Title: Gay Purr−ee
<hidden for brevity>

Wikipedia Title: A Christmas Carol
<hidden for brevity>

Wikipedia Title: UPA (animation studio)
<hidden for brevity>

Q: Mister Magoo's Christmas Carol was produced by the same studio that produced a film that featured the only animated−film role by who?
A: Mister Magoo's Christmas Carol was produced by United Productions of America studio. United Productions of America studio produced a film Gay Purr−ee, which features the voice of Judy Garland in her only animated−film role. So the answer is: Judy Garland.

Wikipedia Title: Lake Wales Medical Center
<hidden for brevity>

Wikipedia Title: Church of the Holy Spirit (Lake Wales, Florida)
<hidden for brevity>

Wikipedia Title: Lake Wales, Florida
<hidden for brevity>

Q: What was the 2014 population of the city where Lake Wales Medical Center is located?
A: Lake Wales Medical Center is located in the city of Polk County, Florida. The population of Polk County in 2014 was 15,140. So the answer is: 15,140.