ThinkSum: Probabilistic reasoning over sets using large language models

Batu Ozturkler
Stanford University
Stanford, California, USA
ozt@stanford.edu

Nikolay Malkin
Mila, Université de Montréal
Montréal, Québec, Canada
nikolay.malkin@mila.quebec

Zhen Wang
Ohio State University
Columbus, Ohio, USA
wang.9215@osu.edu

Nebojsa Jojic
Microsoft Research
Redmond, Washington, USA
jojic@microsoft.com

Abstract

Large language models (LLMs) have a substantial capacity for high-level analogical reasoning: reproducing patterns in linear text that occur in their training data (zero-shot evaluation) or in the provided context (few-shot in-context learning). However, recent studies show that even the more advanced LLMs fail in scenarios that require reasoning over multiple objects or facts and making sequences of logical deductions. We propose a two-stage probabilistic inference paradigm, **ThinkSum**, which reasons over sets of objects or facts in a structured manner. In the first stage (**Think** – retrieval of associations), a LLM is queried in parallel over a set of phrases extracted from the prompt or an auxiliary model call. In the second stage (**Sum** – probabilistic inference or reasoning), the results of these queries are aggregated to make the final prediction. We demonstrate the possibilities and advantages of **ThinkSum** on the BIG-bench suite of LLM evaluation tasks, achieving improvements over the state of the art using GPT-family models on thirteen difficult tasks, often with far smaller model variants. We also compare and contrast **ThinkSum** with other proposed modifications to direct prompting of LLMs, such as variants of chain-of-thought prompting. Our results suggest that because the probabilistic inference in **ThinkSum** is performed outside of calls to the LLM, **ThinkSum** is less sensitive to prompt design, yields more interpretable predictions, and can be flexibly combined with latent variable models to extract structured knowledge from LLMs. Overall, our proposed paradigm represents a promising approach for enhancing the reasoning capabilities of LLMs.

1 Introduction

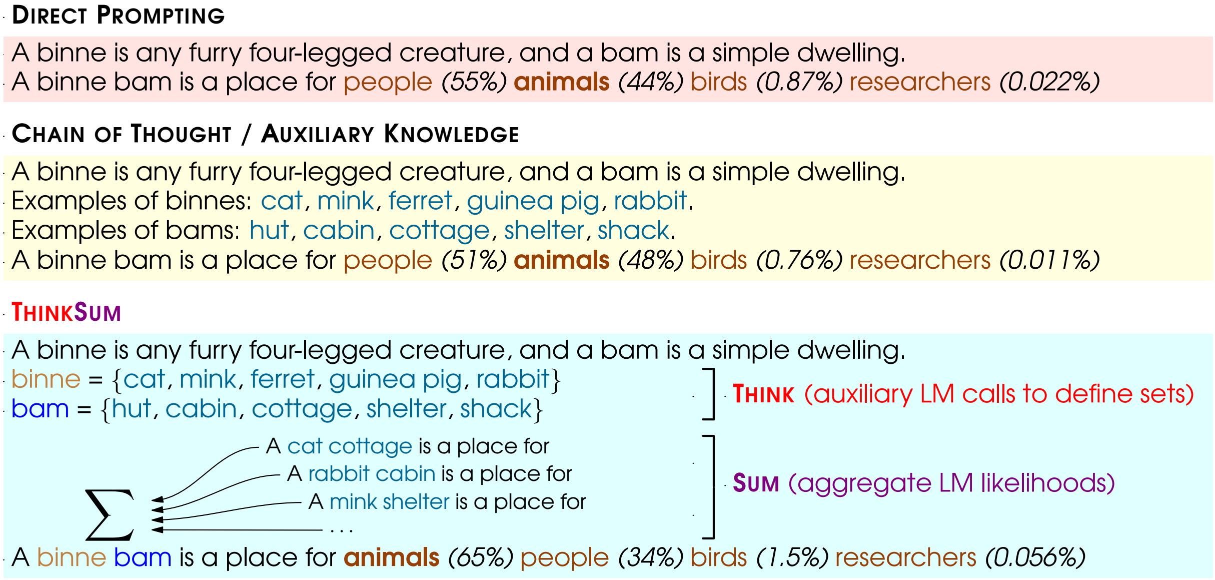
Large language models (LLMs; Brown et al., 2020; Rae et al., 2021; Chowdhery et al., 2022) can recall a broad range of basic facts, recognize and mimic various forms in language, and efficiently extrapolate analogies in structure and meaning. These abilities allow LLMs to excel in zero-shot and few-shot tasks formulated as the generation or selection of a likely completion to a prompt. This formulation requires LLMs to perform **fast associative thinking**, in which each token of text in the sequence making up the answer is generated or scored in one pass through the model and, other than that, no intermediate information is created or retained. This fast thinking is made possible by the compression of information that is repeated in a variety of ways in large training datasets, within the LLM's weights.

However, it is increasingly evident that when **reasoning**, or slow thinking, is required, failure modes of LLMs are revealed. In our usage, reasoning refers to the sequential manipulation of concepts that can be expressed in language. Tasks that require iterative retrieval of rarely stated knowledge, uncertainties over multiple objects or facts, or multiple steps of deduction are difficult even for the most advanced LLMs (Suzgun et al., 2022). In a recently designed suite of evaluations, BIG-bench (Srivastava et al., 2022), some of the tasks where the gap between machine and human performance is large involve inference sequences with nested counterfactuals (Logical deduction), concepts introduced through definitions (Conceptual combinations), etc. (see Fig. B.1). These are tasks where a human solver's intuitive feeling of '(in)coherence' is insufficient to produce the right answer, and a sequence of thoughts, along with the use of intermediate results, may be necessary to arrive at the solution, particularly when working memory is insufficient.

We show several tasks in BIG-bench that can be addressed by a two-component mechanism, which we name **ThinkSum**[1]:

[1] **ThinkSum** is named by analogy with other algorithms

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 1216–1239
July 9-14, 2023 ©2023 Association for Computational Linguistics



DIRECT PROMPTING

A binne is any furry four-legged creature, and a bam is a simple dwelling.
A binne bam is a place for people *(55%)* **animals** *(44%)* birds *(0.87%)* researchers *(0.022%)*

CHAIN OF THOUGHT / AUXILIARY KNOWLEDGE

A binne is any furry four-legged creature, and a bam is a simple dwelling.
Examples of binnes: cat, mink, ferret, guinea pig, rabbit.
Examples of bams: hut, cabin, cottage, shelter, shack.
A binne bam is a place for people *(51%)* **animals** *(48%)* birds *(0.76%)* researchers *(0.011%)*

THINKSUM

A binne is any furry four-legged creature, and a bam is a simple dwelling.
binne = {cat, mink, ferret, guinea pig, rabbit}
bam = {hut, cabin, cottage, shelter, shack}

THINK (auxiliary LM calls to define sets)
A cat cottage is a place for
A rabbit cabin is a place for
A mink shelter is a place for
. . .
$\sum$
SUM (aggregate LM likelihoods)

A binne bam is a place for **animals** *(65%)* people *(34%)* birds *(1.5%)* researchers *(0.056%)*

Figure 1: An example adapted from the CONCEPTUAL COMBINATIONS (INVENTED WORDS) task, in which models must select the most likely completion of a phrase that includes nonce words whose definitions are given. **Top: Direct prompting** evaluates completion likelihoods normalized over the four answer choices ('people', 'animals', 'birds', 'researchers'). **Middle: Chain-of-thought**-like or **auxiliary knowledge** approaches would query a LLM or knowledge base for additional context. This example shows the brittleness entrusting all 'reasoning' to self-attention in linear text, especially in smaller models, which have stronger recency bias (Malkin et al., 2022): if we simply list generated examples as the additional context in the prompt, the recency bias causes the LLM to still give a higher probability to 'people' than to 'animals', simply because 'bam' (simple dwelling) examples are given after the 'binne' examples. **Bottom:** Our **ThinkSum** approach to this task queries a LLM (GPT-2 XL) to produce sets of examples defining the nonce words, then marginalizes over substitutions of these examples into the target phrase.

- **Think** (fast thinking / association / knowledge retrieval step): creating an association of text spans with sets of strings. This process may involve generation from a language model, as is the case in Fig. 1, where the novel word 'binne' is associated with the set of strings {'cat', 'mink', . . . } by prompting GPT-3 with the definition and asking for examples. Alternatively, it may consist solely of a scoring mechanism, resulting in the formation of a matrix of probabilities on which probabilistic inference is performed.
- **Sum** (slow thinking / **Sum**marization / reasoning step): probabilistic inference that aggregates generated strings or probabilities to produce the final answer. Summarization typically involves, and often entirely consists of, summing of probabilities of strings (computed in the **Think** step), as in Fig. 1, where the final word is assumed to be sampled from a mixture of possible substitutions of 'binne' and 'bam' words into the input.

We discuss different ways to **Think** and to **Sum** in section §2, but we start with one example, illustrated in Fig. 1 (bottom), motivated by the CONCEPTUAL COMBINATIONS (INVENTED WORDS) task in BIG-bench. In this task, the LLM is provided with the definitions of two invented words and asked to infer the most plausible sentence that uses a combination of the invented words. As the words are not common or consistently used in the training set, the LLM needs to understand and combine the definitions of the invented words to reason about the meaning of the combination. The LLM is queried to produce example instances of the invented words with the help of the definitions. These example instances can be substituted into the query in place of the invented words. By mapping individual spans of the text of interest to sets, we arrive at a mixture model (in this example, a mixture with 25 components for 5 possible replacements of each word), which can be used in the same manner the original LLM is used, either to score text or to generate it token by token. When we score all candidate completions using this mixture model and normalize over the four choices, the correct answer – that 'binne bams' are for animals and not people – becomes the most likely.

with 'expand' and 'aggregate' steps, such as MapReduce in distributed computing and sum-product in graphical models.

An important difference between our **ThinkSum** and existing chain-of-thought-like prompt engineering methods (Wei et al., 2022; Kojima et al., 2022), is that our reasoning step is not reduced to a generation problem for the LLM, but is performed as a probabilistic inference external to the LLM. This reduces vulnerability to features of the prompt, such as accidental distraction of the LLM by spurious patterns (see Fig. 1, middle). Instead, we engineer the slow thinking process to make parallel calls to the LLM to query for intermediate information, then possibly perform programmatic recombination of strings (**Think**). The final reasoning step – in which likelihoods obtained from the LLM for the recombinations derived from earlier steps of the reasoning process are combined to make the final prediction – is left to classical probabilistic reasoning (**Sum**). In a sense, **Sum** replaces the self-attention mechanism over linear text, which is used as the sole 'reasoning' mechanism in chain-of-thought-like approaches that expect the intermediate 'thoughts' to take the form of generated tokens intervening between the input and output.

Imposing an alternative reasoning system over an associative "knee-jerk reaction" system has an analogy with models of human cognitive processes (Tversky and Kahneman, 1974; Kahneman, 2011) that separate System 1 (fast thinking) and System 2 (slow thinking). System 2 acts as a 'controller' that can prime System 1 to appropriately bias its fast thinking. In the context of reasoning with deep learning models, System 2 has been interpreted as operating with sparse concepts that can be described in language (Bengio, 2017; Goyal and Bengio, 2020). Through repeated usage, the functions of System 2 become compressed into System 1 intuitions, in the same manner that iterative 'reasoning' functions of which smaller LLMs are not capable become zero-shot generation capacities for large LLMs. As is the case with humans, there is always the next frontier of problems where a trained model with remarkable 'intuition' needs to be slowed down. The main claim of this paper is that more is possible with LLMs of existing scale when they are used in concert with a wise controller that allows for probabilistic inference.

2 ThinkSum

2.1 How to Think

Here we list examples of the "fast thinking" that precedes the summarization stage.

Elementary string manipulations. Standard ways to turn a question into a prompt that can be given to a LLM for generation or scoring involve choices (e.g., of the prompt format) that can be seen as being made by a controlling agent. The default approach to multiple-choice questions is to write them as Cloze tasks. However, there are nontrivial operations used in inference procedures that sometimes work better, such as:

- **Order inversion**: Exchanging the order of the question and answers, as in Min et al. (2022).
- **Premise erasure**: Deleting a part of the question. Removing a premise with which the answer is expected to have high mutual information is a step in inference procedures that aim to correct for bias towards answers with high unconditional likelihood (Zhao et al., 2021; Holtzman et al., 2021; Malkin et al., 2022).

Substitution and normalization. An example is shown in Fig. 1. Elements from a set may be substituted in place of 'slot' words in a prompt, such as 'cat' substituted for 'binne' in the prompt "`A binne bam is a place for`". This operation can be combined with syntax-normalization steps that are reliably achieved by standard NLP tools, such as ensuring subject-verb agreement.

Example and list generation. A LLM can be prompted to generate or score lists of words or phrases. We suggest and experiment with three instances of this:

- **Example generation**: In Fig. 1, the LLM is prompted to turn a definition or characterizing property, such as 'simple dwelling', into a list of examples. This can be achieved with a prompt such as "`A bam is a simple dwelling. Examples: 1.`". The generated completion can be parsed into a set to be used later in the inference procedure.
- **List extension**: A similar approach can also be used to hallucinate additional possible answers to questions, as we will show in some of the experiments.
- **List of words**: Similar prompts provide an even simpler **Think** method that we use for scoring – but not generation – in several tasks. Just prompting a LLM with "`List of words:` A, B", where A and B are words or phrases, and computing the likelihood of B conditioned on "`List of words:` A," is a good measure of semantic relatedness of A and B.

Fact generation. This way of **Think**ing associates an input word with a set of phrases in a similar manner to generating examples from a definition. It can be achieved with prompts such as "`List facts about cats. 1.`" The generated facts are good targets for substitutions of other concepts ('dogs', 'galaxies') in place of the concept ('cats') about which facts are generated. A variation on this asks the LLM to generate differences between two concepts, as shown in Fig. 2 (right).

Translation. The LLM can be prompted to convert between different forms of representing the same concept as a sequence of tokens. We use two basic examples of this in experiments:

- Translation between languages by prompting the LLM in formats such as "`French: J'adore les chats noirs. English:`". A very similar approach can be used to convert non-alphabetic symbols, such as emoji, into words with similar meanings.
- Converting text to formal (symbolic) structures, like turning a word problem into a collection of mathematical equations.

2.2 How to Sum

Elementary inference. As above, we begin by listing existing standard ways of turning LLM outputs into answers, which we see as trivial cases of aggregation (**Sum**).

- **Majority/minority vote (argmin/argmax)**: a component of most answer selection procedures.
- **Ratio of likelihoods**: Likelihoods from different variants of the same prompt can be combined by considering their ratio or more general log-linear or other mixture. For example, this can be done to correct the likelihood of an answer conditioned on a question by its unconditional likelihood, in combination with the **Premise erasure** operation described above.

Mixture (average) aggregation. A collection of prompts can be treated as the components of a mixture model over completions. An example is shown in Fig. 1, where substitutions of a set of words yield 25 different prompts. Likelihoods of the completion over these 25 prompts are averaged.

Product aggregation. We use products of likelihoods in two different ways:

- In a similar way as mixtures, but when the more natural probabilistic model has *all* elements of a set (of prompts) generating the answer, such as when a description or definition must be satisfied by all concepts in a set.
- In a task where we are to determine whether a statement S or its negation S' is true, we can compute the likelihood of both S and S' being true (as posterior over the tokens 'True' and 'False' in an appropriate prompt), then compare $p(\texttt{True}|S)p(\texttt{False}|S')$ (S is true and S' is false) with $p(\texttt{False}|S)p(\texttt{True}|S')$ (S is false and S' is true).

3 Experiments

In this section, we perform case studies on three tasks from the BIG-bench suite to demonstrate the possibilities of the inference approaches discussed in §2. We also experiment with ten other tasks from BIG-bench; the best results are summarized in Table 1 and the methods, grouped by the style of **Think**ing and **Sum**ming, are described in Appendix (§A).

All details of the tasks can be found in the Appendix (§C). Comparisons to direct prompting and algorithms that append retrieved or generated tokens to the prompt are given in §3.4.

3.1 Conceptual combinations: Invented words

In INVENTED WORDS, two nonce words x_1, x_2 are defined and the correct statement must be chosen out of a set of statements $S = \{s_j\}$ that begin with (possibly inflected forms of) "x_1 x_2" (Fig. 1).

We use an **Example generation** prompt to obtain a set of example words fitting the definitions of x_1 and x_2. We thus obtain sets S_1 and S_2 of words that can be substituted for x_1 and x_2, respectively.

We treat each statement s_j as a template into which words $w_1 \in S_1$ and $w_2 \in S_2$ can be substituted by replacing x_i with w_i and normalizing the syntax to ensure subject-verb agreement. Denoting by $s_j\langle w_1, w_2\rangle$ such a substitution, we form a vector of probabilities p_j by scoring the **Substitution** of each possible pair of words into each statement and performing **Mixture aggregation** and considering the **Ratio of likelihoods** with the template without substitution:

$$p_j = \frac{\frac{1}{|S_1||S_2|}\sum_{w_1\in S_1, w_2\in S_2} p_{\mathrm{LLM}}(s_j\langle w_1, w_2\rangle)}{p_{\mathrm{LLM}}(s_j)}.$$

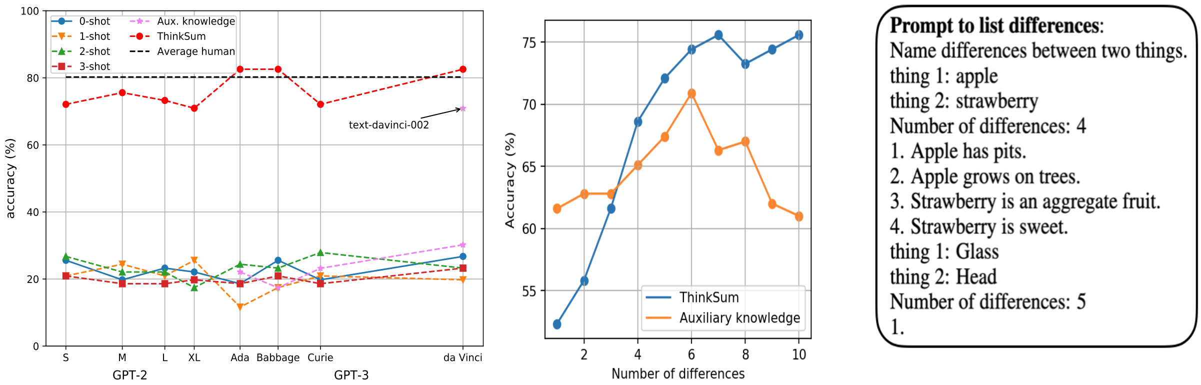
The statement s_j with highest likelihood under this normalized mixture, $\arg\max_j p_j$, is selected.

3.2 Odd one out

We examine possible **Think** and **Sum** approaches in depth on the ODD ONE OUT task, in which the

Task	Avg. H	GPT-3 (davinci) n-shot $n = 0$	1	2	3	**ThinkSum** GPT-3	InstructGPT	GPT-2 XL
INVENTED WORDS (§3.1)	N/A	0.29	0.14	0.14	0.21	0.64	**0.71**	0.29
ODD ONE OUT (§3.2)	0.80	0.27	0.20	0.23	0.23	0.80	**0.84**	0.71
FIVE OBJECTS (§3.3)	N/A	0.23	0.29	0.28	0.32	–	**0.77**	–
SPORTS UNDERSTANDING (§A.1)	0.71	0.50	0.50	0.50	0.50	0.71	**0.74**	0.54
KNOWN UNKNOWNS (§A.1)	**0.80**	0.61	0.52	0.48	0.50	0.54	0.76	–
MISCONCEPTIONS RUSSIAN (§A.2)	0.65	0.33	0.33	0.41	0.35	**0.70**	0.61	–
EMOJI MOVIE (§A.2)	**0.93**	0.12	0.18	0.12	0.19	0.80	0.75	–
PARSINLU READING COMPREHENSION (§A.2)	0.02	0.00	0.00	0.00	0.00	–	0.02	–
PHRASE RELATEDNESS (§A.3)	0.74	0.37	0.42	0.52	0.59	0.85	**0.87**	0.79
CODENAMES (§A.3)	0.18	0.01	0.11	0.16	0.19	0.37	**0.41**	0.36
NOVEL CONCEPTS (§A.4)	0.67	0.47	0.47	0.56	0.56	0.72	**0.75**	0.50
CODE LINE DESCRIPTION (§A.4)	0.60	0.32	0.32	0.28	0.32	0.83	**0.90**	0.77
LANGUAGE IDENTIFICATION (§A.5)	0.16	0.16	0.12	0.13	0.11	**0.57**	–	0.30

Table 1: Standard metric (BLEU for CODENAMES, accuracy for other tasks) for GPT-3 175B (davinci) and **ThinkSum** with 175B (davinci), InstructGPT and GPT-2 XL on BIG-bench tasks. A '–' indicates that the model and task combination was not evaluated because the model does not reliably execute the appropriate **Think** prompt. We did not evaluate InstructGPT on LANGUAGE IDENTIFICATION due to the large dataset size and API quota.



Prompt to list differences:
Name differences between two things.
thing 1: apple
thing 2: strawberry
Number of differences: 4
1. Apple has pits.
2. Apple grows on trees.
3. Strawberry is an aggregate fruit.
4. Strawberry is sweet.
thing 1: Glass
thing 2: Head
Number of differences: 5
1.

Figure 2: ODD ONE OUT. **Left:** Performance of GPT-3 (n-shot, $n = 0, 1, 2, 3$), auxiliary knowledge, and **ThinkSum** with various model sizes. **Middle:** Auxiliary knowledge vs. **ThinkSum** with varying number of differences. **Right:** Prompt used to generate knowledge statements.

word in a set $W = \{w_i\}$ that is *least* semantically related to the others must be chosen (e.g., *Pick the odd word out: glass, head, arm, leg, hand, foot*).

List of words. We form a semantic relatedness matrix P_{ij} by querying the LLM with a **List of words Think** prompt for each pair of indices i, j:

$$P_{ij} = p_{\text{LLM}}(w_j \mid \text{“}\texttt{List of words: } w_i\texttt{, }\text{”}).$$

This matrix is aggregated by averaging over j (in log domain) and selecting the i with lowest average, i.e., least likelihood of being generated by a product mixture of all words in the set: $i = \arg\min_i \prod_j P_{ij}$. This is a case of **Product aggregation**.

Because this approach is the most successful with all model sizes we experimented with, its performance is reported in Table 1. Remarkably, near-average-human accuracy is maintained for all model sizes from GPT-2 Small to the largest GPT-3 model (Fig. 2 (left)).

Fact generation. As an alternative approach, we use a **Fact generation** prompt. An effective way to mine facts for semantic relatedness tasks is to consider two items in the same context in order to get relevant facts regarding how items are related to each other (prompt in Fig. 2 (right)). The demonstration used in the prompt ensures that the LLM generates statements in an expected format, which can be parsed and used for probability computation later. Using this prompt, we obtain a collection of statements $S = \{s_i\}$ about items w_j. We treat each generated s_i as a template into which different words w can be substituted and denote by $s_i\langle w\rangle$ the **Substitution** of word w into template s_i. We then form a $|S| \times |W|$ matrix P_{ij}, defined by $P_{ij} = p_{\text{LLM}}(s_i\langle w_j\rangle)$. Then, we can perform **Minority voting**: we take argmin over j and pick as the answer the most frequently occurring value, i.e., the item that is most often the least likely to fit a generated statement.

Comparison with auxiliary knowledge approaches. We compare our method with a knowledge-based prompting method, herein referred to as auxiliary knowledge. In auxiliary knowledge, we prepend generated facts in the prompt before the question. Details of the prompt for auxiliary knowledge are provided in §D.3. In Figure 2 (middle), we show that the accuracy of **Fact generation**-based **ThinkSum** rises as the number of generated facts is increased, while the auxiliary knowledge technique peaks and then degrades as the prompt lengthens.

Fig. 2 (left) shows how performance varies with the size of the LLM used for GPT-3, auxiliary knowledge and **ThinkSum** on ODD ONE OUT. Even with GPT-2 Small, **ThinkSum** dramatically improves over much larger largest zero- or few-shot models with or without auxiliary knowledge. A finetuned iteration of the largest GPT-3 model, text-davinci-002, is the only model variant that, with the help of auxiliary knowledge, achieves competitive performance with **ThinkSum**. This result provides experimental evidence for our claim that while new models may create qualitative jumps, **ThinkSum** can push the performance limits of smaller models.

Latent variable models. As we have shown, the detection of the odd item can be performed with simple inference operations on items, facts, and their joint likelihoods. However, it is also possible to assume a latent structure in the items and facts, consisting of two or more clusters such that the facts and items belonging to a cluster can be freely interchanged. We describe a problem-specific latent variable model that enables selecting the facts that characterize the majority class, thus explaining why the minority item is ruled as the odd one out and helping interpret the decisions of the system.

We model items $i \in I$ and facts $f \in F$ as being generated from a latent class $c \in \{0, 1\}$. The distribution is modeled as:

$$P(i, f) = \sum_{c} P(c)P(i|c)P(f|c)$$

where $P(i, f)$ is a matrix of likelihoods from the LLM and the semantic components, groupings $P(i|c)$ and $P(f|c)$, are derived from the matrix using a standard iterative expectation-maximization (EM; Dempster et al., 1977) inference procedure (see §E). Then, the score for an item i belonging to a cluster and all other items $m \in S, \{m \neq i\}$ belonging to another cluster can be found as $S_i = \sum_{c,c' \neq c} P(i|c)P(c) \prod_{m \neq i} P(m|c')P(c')$.

Model	LoW	LVM	MV
text-davinci-002	0.84	0.67	0.70
text-davinci-001	0.74	0.77	0.70

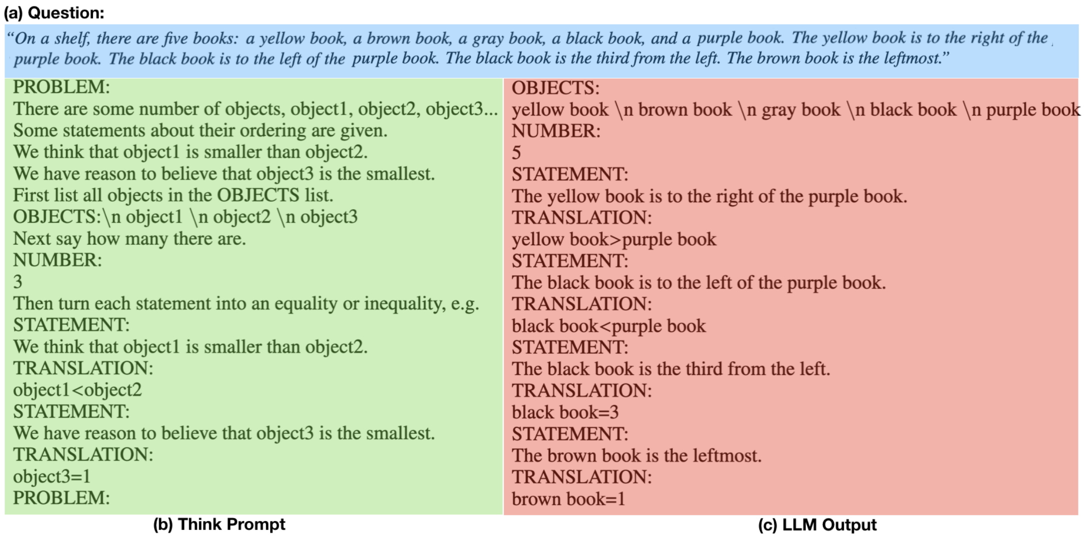
Table 2: Different alternatives of probabilistic reasoning with **ThinkSum** for solving ODD ONE OUT: list of words, latent variable model, minority voting.

We show the effectiveness of the latent variable models in Table 2, where we analyze different methods for solving ODD ONE OUT using the InstructGPT variants text-davinci-001 and text-davinci-002. For the 'latent variable model' and 'minority voting' methods, we use number of differences $N_d = 5$. The latent variable model is trained for 200 EM iterations. All probabilistic reasoning methods perform well, outperforming previous baselines reported in Table 1. Inference using EM, as well as the other approaches, can be seen as a **Sum** (inference) operation and can be applicable in other tasks of similar structure.

3.3 Logical deduction

In the LOGICAL DEDUCTION task, different types of items and clues regarding their order are provided (Fig. 3(a)). The goal is to select the correct statement from a set of statements about their placements. The ordering problems involve different types of objects (cars, birds, etc.) and orderings (by size, price, contest ranking, etc.). The task creators emphasize that this task requires parsing information about multiple objects and their relationships, understanding rules regarding ordered objects in various scenarios, and iteratively applying these rules. The LLM calls in the **Think** stage of **ThinkSum** can perform mappings required to parse information and understand rules, and the **Sum** stage can integrate mappings of objects to the placements under these rules. Here, we use a **Translation** prompt to map the given problem into a set of mathematical (in)equalities (Fig. 3(c)).

The **Translation** prompt in Fig. 3(b), containing generic ordering statements and object names that are not used in the task as an in-context demonstration, is sufficient to perform the translation from natural language to equations. By prepending this



(a) Question:

"On a shelf, there are five books: a yellow book, a brown book, a gray book, a black book, and a purple book. The yellow book is to the right of the purple book. The black book is to the left of the purple book. The black book is the third from the left. The brown book is the leftmost."

```
PROBLEM:
There are some number of objects, object1, object2, object3...
Some statements about their ordering are given.
We think that object1 is smaller than object2.
We have reason to believe that object3 is the smallest.
First list all objects in the OBJECTS list.
OBJECTS:\n object1 \n object2 \n object3
Next say how many there are.
NUMBER:
3
Then turn each statement into an equality or inequality, e.g.
STATEMENT:
We think that object1 is smaller than object2.
TRANSLATION:
object1<object2
STATEMENT:
We have reason to believe that object3 is the smallest.
TRANSLATION:
object3=1
PROBLEM:
```

(b) Think Prompt

```
OBJECTS:
yellow book \n brown book \n gray book \n black book \n purple book
NUMBER:
5
STATEMENT:
The yellow book is to the right of the purple book.
TRANSLATION:
yellow book>purple book
STATEMENT:
The black book is to the left of the purple book.
TRANSLATION:
black book<purple book
STATEMENT:
The black book is the third from the left.
TRANSLATION:
black book=3
STATEMENT:
The brown book is the leftmost.
TRANSLATION:
brown book=1
```

(c) LLM Output

Figure 3: Details for LOGICAL DEDUCTION. (a) Example question from the task, (b) demonstration for the **Think** prompt, (c) example LLM output. The demonstration induces the LLM to generalize from generic objects ordered by size to books ordered by position.

demonstration prompt to a problem statement, we induce the LLM to map the objects in the problem to the set of strings corresponding to numbers from 1 to N, where N is the number of objects, and to produce a set of inequalities (Fig. 3(c)).

Once a translation of the problem into a set of inequalities is obtained, the **Sum** stage considers all possible mappings of items to indices to determine the mapping compatible with the discovered set of (in)equalities. This can be done by an external algorithm or by the LLM itself, as an LLM may be capable of understanding that, for example, "2>3" is a less likely string than "2>1" (see §D.2).

Finally, the probability of each of the candidate statements, like "`yellow book=2`", can thus be obtained by:

$$p(\text{“}\texttt{yellow book=2}\text{''} \mid T) \propto \sum_{\mathbf{b}\in\{1,\dots,N\}^N} p_{\text{LLM}}(\{T_t\langle\mathbf{b}\rangle : T_t \in T\} \cup \{\text{“}\texttt{yellow book=2}\text{''}\langle\mathbf{b}\rangle\}) \quad (1)$$

where $\mathbf{b}$ denotes the vector of positions for the N items (e.g., $(5,2,3,4,1)$), $T = \{T_t\}_{t=1}^{N}$ is the set of inequalities obtained from the **Translation** prompt as a set of strings (e.g., "`black book<purple book`"), and $s\langle\mathbf{b}\rangle$ denotes the substitution of the corresponding entry in $\mathbf{b}$ in place of the object name in the string s (e.g., "4<5"). The term inside the sum is a case of **Product aggregation**: the LLM likelihoods of all strings in the set are multiplied.

In summary, our solution to this task involves composition of two **Think** operations – a **Translation** into a set of equations and then **Substitution** of numbers in place of item names – and two **Sum** operations – a **Product aggregation** followed by a **Mixture aggregation**. (Other options are discussed below.)

Results and discussion. For the 500 LOGICAL DEDUCTION problems with $N = 5$ objects, **ThinkSum** yields an accuracy of 77% (see Table 1), besting the average human performance. When the necessary summations become large, it becomes very unlikely that pure prompt engineering can be competitive, as even humans need paper and pencil to create and attend to many alternative solutions, and would likely translate the premises into a simpler notation using a single letter (representing a variable to which a numeric value can be assigned) to represent each object, rather than directly attending to the words in the problem statement.

We also test an auxiliary knowledge method akin to chain-of-thought reasoning, where the information obtained with the prompt in Fig. 3 is appended to the LLM input. In particular, the problem, together with its translation into inequalities, is used as a prompt to each of the answer options, and then the option with the highest likelihood is chosen for the answer. This approach does improve over straightforward zero-shot GPT-3 scoring, but only raises the accuracy to 50% (see §3.4 and Table 3).

Optimizations, failure modes, and extensions. We have seen that InstructGPT is able both to translate logical deduction problems into (in)equalities

(Fig. 3) and to evaluate each of them after replacement of items with position numbers (§D.2). We conclude that the **Sum** stage is there simply to *search* over all possible mappings, the way a human might. But, just as a human might use shortcuts in the search, the **Sum** stage of **ThinkSum** could be implemented in more or less efficient ways. For example, instead of summing over all possible assignments of the five items, we can avoid the ones that are not permutations of $\{1, 2, 3, 4, 5\}$. Furthermore, instead of using p_{LLM} from Fig. D.1 in (1), we can simply evaluate each inequality externally, giving a high constant probability for each inequality $T_t\langle \mathbf{b} \rangle$ that is true and a low probability when it is false, or the summing can be aborted whenever an incorrect statement is detected in a particular assignment **b** of positions to items.

The prompt in Fig. 3(b) instructs the LLM to assign positive integers depending on the language used (e.g., the smallest object gets 1), but a common behaviour of the LLM is to generalize to assigning negative numbers, such as using -2 to represent 'second from the end' (or second-largest, etc.). To remain robust to such a behavior of the **Think** stage, we can convert negative position numbers r into $N + r + 1$ before evaluating statements. However, a persistent failure mode of this kind of **ThinkSum** is that the LLM may translate inequality statements inconsistently with equality statements (e.g., by coding the leftmost item as 1 and being consistent with this choice for other equality constraints, but translating inequality constraints consistently with the reverse order, with 'left of' meaning $>$). Such failures can be addressed by careful engineering in the **Sum** stage, such as by summing out a binary latent variable indicating whether inequalities should be reversed. This increases the number of model evaluations, but also allows for robust auto-correction by the **Sum** stage of inconsistencies in the **Think** stage.

3.4 Comparisons with chain-of-thought and auxiliary knowledge approaches

ThinkSum vs. auxiliary knowledge. Table 3 shows the comparison of **ThinkSum** with algorithms that append auxiliary knowledge as an oracle 'reasoning chain'. For PHRASE RELATEDNESS, auxiliary knowledge was generated using the "list differences" prompt shown in Fig. 2 (right). For both auxiliary knowledge and **ThinkSum**, 6 generated differences were used, as that was the best for auxiliary knowledge (see Fig. 2 (middle)). **ThinkSum** ODD ONE OUT and PHRASE RELATEDNESS are solved with the "list of words" prompt. For LOGICAL DEDUCTION, the **Think** prompt shown in Fig. 3 was included before the question in the prompt. In all cases, **ThinkSum** outperforms auxiliary knowledge.

	ODD ONE OUT	PHRASE RELATEDNESS	LOGICAL DEDUCTION ($N = 5$)
ThinkSum	0.84	0.87	0.77
Aux. knowledge	0.71	0.75	0.50

Table 3: **ThinkSum** vs. auxiliary knowledge with text-davinci-002.

ThinkSum vs. chain of thought. Following Wei et al. (2022), we use "chain-of-thought (CoT) methods" to mean LLM scoring approaches that use insertion of generated tokens between the prompt and the target answer. The model is taught, using few-shot demonstrations, how to generate these intermediate tokens. Above we have compared **ThinkSum** with approaches that add *extracted* (from an auxiliary LM call), not *generated* (within the LM's linear workspace) token sequences after the prompt, for the ODD ONE OUT, PHRASE RELATEDNESS, and LOGICAL DEDUCTION tasks (see Table 3).

With suitable examples, it may be possible for a CoT approach to replace the **Think** phase, by learning from demonstrations to generate the appropriate knowledge, and parts of the **Sum** phase, although inference over parallel evaluations of the LLM is no longer possible. Our auxiliary knowledge baselines make precisely that generous assumption and focus the comparisons on the need for parallel calls and reasoning over possibilities using probabilistic inference (instead of leaving it to the LLM to make the right conclusions from the list of extracted alternatives).

Although we expect that appending facts in a standard format to the prompt would help the model more than teaching the model to generate these facts, we experimented with CoT approaches on several tasks. Table A.1 shows example demonstrations and prompt formats used for each task, and Table 4 shows the results using two variants of the largest GPT-3 model.

As expected, **ThinkSum** outperforms CoT prompting on all tasks with all variants except KNOWN UNKNOWNS with the davinci variant, where direct prompting already performs well. (We did not evaluate **ThinkSum** with davinci on LOGICAL DEDUCTION because prompts like the one

	GPT-3 (davinci)			GPT-3 (davinci-002)	
Task	Direct	CoT	**ThinkSum**	CoT	**ThinkSum**
Odd one out	0.27	0.33	0.80	0.64	0.84
Phrase relatedness	0.59	0.55	0.85	0.79	0.87
Logical deduction	0.32	0.25	–	0.39	0.77
Known unknowns	0.61	0.70	0.54	0.74	0.76
Invented words	0.29	0.50	0.64	0.64	0.71

Table 4: Comparison of **ThinkSum** with chain-of-thought prompting approaches.

in Figure 3 did not reliably produce outputs in the correct format; notice that CoT is barely better than random guessing (20%).)

When interpreting these results, it is important to note that only one prompt format was evaluated for both CoT and **ThinkSum**, and the format of prompts and demonstrations can have a strong and often unpredictable effect on the LLM. We observed that CoT approaches are highly sensitive to minor changes in the prompt format or the construction of in-context examples, consistent with the known biases of in-context learning (Lu et al., 2022; Zhao et al., 2021). On the other hand, using structured, shorter components is more reliable, as demonstrated by the efficacy of the **Think** prompts used in **ThinkSum**.

4 Related work

Improvements to LLM inference. After the discovery of the in-context learning abilities of LLMs, there has been an explosion of interest in improving inference with LLMs in the zero-shot and few-shot setting (Brown et al., 2020; Chowdhery et al., 2022; Rae et al., 2021). One approach to improving the reasoning abilities of LLMs involves appending, or learning to generate, auxiliary knowledge within the prompt (Shwartz et al., 2020; Zelikman et al., 2022; Nye et al., 2021a). Recently, more general auxiliary knowledge or chain-of-thought prompting methods have been proposed (Wei et al., 2022; Wang et al., 2022b; Zhou et al., 2022a; Creswell et al., 2022; Wang et al., 2022a; Liu et al., 2022b), including those that allow a control flow external to the main LLM (Khot et al., 2022). Later, Kojima et al. (2022) showed zero-shot chain-of-thought prompting can improve performance on a variety of reasoning tasks. This method does not require any hand-crafted few-shot examples, which is a shared property with **ThinkSum**. (Nye et al., 2021b) observed that a dual-system approach where an associative "System 1" and a logical "System 2" can increase coherence of LLMs in tasks such as robust story generation and grounded instruction following. The two-step paradigm in **ThinkSum** is similar, where "System 1" is the (querying of the LLM for) fast thinking, and "System 2" is the probabilistic inference step.

Brittleness of chain-of-thought prompting. Despite the recent success of chain-of-thought approaches, recent studies have raised concerns regarding the limitations of chain-of-thought approaches. Webson and Pavlick (2022) observed that instructive prompts perform similarly with misleading or intentionally irrelevant prompts. Additionally, Ye and Durrett (2022) showed improvements due to few-shot chain-of-thought are not observed in question answering, or natural language inference. More critically, few-shot prompts are highly sensitive to the order in which the samples are provided, the prompt format, and the selection of in-context examples, (Lu et al., 2022; Zhao et al., 2021). Thus, it is crucial to design techniques that are robust to such changes in the prompt.

Inference as reasoning. Iterative inference over LLM outputs has been proposed for tackling true/false question answering and commonsense question answering (Jung et al., 2022; Liu et al., 2022a). Xie et al. (2021) presents a Bayesian inference perspective on in-context learning, and Dohan et al. (2022) formalizes and unifies existing prompting techniques in a probabilistic framework. Our work generalizes such approaches to perform arbitrary probabilistic inference outside of the LLM.

5 Conclusion

In this paper we presented **ThinkSum**, a two-step probabilistic inference paradigm that reasons over sets in a structured manner. The fast thinking stage of **ThinkSum** allows elementary string manipulations as well as natural language prompting, which may enable numerous approaches to solve a natural language task. Even with far smaller model variants, **ThinkSum** achieves state-of-the-art results on ten difficult tasks in BIG-bench using GPT-family models. The two-step paradigm allows operating over sets instead of manipulating the prompt itself, preventing sensitivity to prompt format during the probabilistic inference in **ThinkSum**, which is performed outside of calls to the LLM. As a result, **ThinkSum** is more robust to prompt design, yields more interpretable predictions, and can be combined with many probabilistic inference approaches to tackle a diverse set of tasks.

Acknowledgments

The authors thank Alexandros Graikos, Sudha Rao, and Alessandro Sordoni for valuable discussions.

Limitations

Our proposed **ThinkSum** has demonstrated strong performance on thirteen challenging BIG-bench tasks. However, it is important to acknowledge certain limitations of the system.

Firstly, as the number of objects or facts that are reasoned over increases, the computation cost will also rise. However, increasing the number of objects will also make the task harder, and direct prompting may cease to work at all (as we indeed observe in BIG-bench results, such as LOGICAL DEDUCTION with more than five objects), while **ThinkSum** offers a generalizable methodology, as the atomic **Think** operations do not increase in complexity as the number of objects grows.

Secondly, when solving a new task, it is necessary to expend human effort to select specific operations in each step, as outlined in §2. This limitation is shared with prompt engineering of all kinds, including direct or chain-of-thought prompting: finding a prompt for a new task requires an often-cumbersome prompt engineering procedure. We have described **ThinkSum** as a general two-stage paradigm, with an external inference step. This generality aims to facilitate the adaptation of **ThinkSum** to new tasks, with minimal modifications to the **Think** and **Sum** steps. Work on automating the prompt engineering procedure (Zhou et al., 2022b) is a promising path towards overcoming this limitation. An alternative to prompt engineering that does not require such human effort is tuning (i.e., differentiable end-to-end learning) of prompts or model parameters; however, this remains impractical for GPT-3-scale models, and attempts to tune models directly on symbolic reasoning chains have met with limited success (Kassner et al., 2020).

Last but not least, **ThinkSum** has mainly been evaluated with GPT-3 (davinci) and InstructGPT (text-davinci-002) models. To further improve performance, it may be beneficial to apply **ThinkSum** to more recent instruction-tuned models such as Flan-PaLM (Chowdhery et al., 2022; Chung et al., 2022), text-davinci-003, ChatGPT, and GPT-4, which seem more capable of robustly performing **Think** steps.

Ethics and impact statement

We foresee no direct or immediate societal impacts arising from this work. However, we would like to emphasize that relying solely on LLMs' associative reactions to prompts can lead to undesired bias in the behaviour of systems. Control of LLMs' reasoning in the way we have proposed can potentially mitigate such bias, due both to the decomposition of the argumentation process into interpretable fact-retrieval steps and to the averaging effect of smoothing out spurious triggers when aggregating many hypotheses and reasoning chains.

References

Yoshua Bengio. 2017. The consciousness prior. *arXiv preprint arXiv:1709.08568.*

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. 2020. Language models are few-shot learners. *Neural Information Processing Systems (NeurIPS).*

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

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

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

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–38.

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. 2022. Language model cascades. *arXiv preprint arXiv:2207.10342.*

Nouha Dziri, Andrea Madotto, Osmar Zaïane, and Avishek Joey Bose. 2021. Neural path hunter: Reducing hallucination in dialogue systems via path grounding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2197–2214, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Anirudh Goyal and Yoshua Bengio. 2020. Inductive biases for deep learning of human cognition. *arXiv preprint arXiv:2011.15091*.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7038–7051, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Daniel Kahneman. 2011. *Thinking, fast and slow*. Macmillan.

Nora Kassner, Benno Krojer, and Hinrich Schütze. 2020. Are pretrained language models symbolic reasoners over knowledge? In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 552–564, Online. Association for Computational Linguistics.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. 2022a. Generated knowledge prompting for commonsense reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3154–3169, Dublin, Ireland. Association for Computational Linguistics.

Tianyu Liu, Yizhe Zhang, Chris Brockett, Yi Mao, Zhifang Sui, Weizhu Chen, and Bill Dolan. 2021. A token-level reference-free hallucination detection benchmark for free-form text generation. *arXiv preprint arXiv:2104.08704*.

Zihan Liu, Mostofa Patwary, Ryan Prenger, Shrimai Prabhumoye, Wei Ping, Mohammad Shoeybi, and Bryan Catanzaro. 2022b. Multi-stage prompting for knowledgeable dialogue generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1317–1337, Dublin, Ireland. Association for Computational Linguistics.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Nikolay Malkin, Zhen Wang, and Nebojsa Jojic. 2022. Coherence boosting: When your pretrained language model is not paying enough attention. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8214–8236, Dublin, Ireland. Association for Computational Linguistics.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Noisy channel language model prompting for few-shot text classification. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5316–5330, Dublin, Ireland. Association for Computational Linguistics.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2021a. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.

Maxwell Nye, Michael Tessler, Josh Tenenbaum, and Brenden M Lake. 2021b. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. *Neural Information Processing Systems (NeurIPS)*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Neural Information Processing Systems (NeurIPS)*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training Gopher. *arXiv preprint arXiv:2112.11446*.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4615–4629, Online. Association for Computational Linguistics.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Amos Tversky and Daniel Kahneman. 1974. Judgment under uncertainty: Heuristics and biases: Biases in judgments reveal some heuristics of thinking under uncertainty. *Science*, 185(4157):1124–1131.

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

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

Albert Webson and Ellie Pavlick. 2022. Do prompt-based models really understand the meaning of their prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States. Association for Computational Linguistics.

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

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. 2021. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*.

Xi Ye and Greg Durrett. 2022. The unreliability of explanations in few-shot in-context learning. *arXiv preprint arXiv:2205.03401*.

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

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. *International Conference on Machine Learning (ICML)*.

Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Francisco Guzmán, Luke Zettlemoyer, and Marjan Ghazvininejad. 2021. Detecting hallucinated content in conditional neural sequence generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1393–1404, Online. Association for Computational Linguistics.

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

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022b. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

A Additional tasks

Descriptions of all the tasks studied here can be found in §C.

A.1 Uncertainty and hallucination detection

LLMs are prone to generating hallucinations that contain incorrect statements. The likelihoods of these statements are often dominated by short plausible patterns, which also makes it difficult for LLMs to evaluate their own uncertainty about a fact. Thus, detection (Liu et al., 2021; Zhou et al., 2021) and reduction of such hallucinations is crucial for widespread use of LLMs in real applications (Dziri et al., 2021; Shuster et al., 2021).

A.1.1 Sports understanding

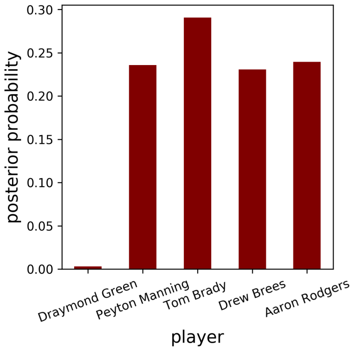


Figure A.1: Example posterior probabilities generated from text-davinci-002 for SPORTS UNDERSTANDING with the description *"threw a touchdown"*. The basketball player given in the question *Draymond Green* has a much lower posterior probability than the generated football players, from which we conclude the sentence *"Draymond Green threw a touchdown."* is implausible.

Questions in SPORTS UNDERSTANDING ask to determine whether it is 'plausible' or 'implausible' that a professional sports player x (e.g., 'Draymond Green', a basketball player) performed an action a associated with a sport (e.g., 'threw a touchdown', an action in American football). It is implied that the combination of x and a is plausible if the sport with which player x is associated coincides with the sport in which action a is performed. We consider an approach that does not rely on identifying the latent variable (sport) as an intermediate step and is thus more generalizable to other domains.

We use an **Example generation Think** prompt to produce a set S of players who perform action a, then do **Posterior computation** by normalizing the likelihood assigned by the LLM to each player in S, as well as x, performing action a:

$$\forall y \in S \cup \{x\} \quad p(y|a) = \frac{p_{\text{LLM}}(\text{``}y\ a\text{''})}{\sum_{y' \in S \cup \{x\}} p_{\text{LLM}}(\text{``}y'\ a\text{''})}$$

The statement is considered to be implausible if the posterior on x is sufficiently low (**Thresholding**) – see Fig. A.1.

A.1.2 Known unknowns

Questions in the KNOWN UNKNOWNS task ask to determine whether the answer to a question is a certain precise concept or 'unknown'.

Given a question q (e.g., "What was the temperature in Cuzco on the day of the Emperor Vespasian's birth") and the candidate precise answer a (e.g., 25°C), we use a **List extension** prompt to generate a set S of other possible answers to q. We then do a **Posterior computation** over S and the original answer a, similar to that used for SPORTS UNDERSTANDING:

$$\forall y \in S \cup \{a\} \quad p(y|q) = \frac{p_{\text{LLM}}(\text{``}q?\ \ y\text{''})}{\sum_{y' \in S \cup \{a\}} p_{\text{LLM}}(\text{``}q?\ \ y'\text{''})}.$$

The answer a is chosen if the posterior on a is sufficiently high (**Thresholding**), and otherwise 'unknown' is chosen.

A.2 Translation between languages and writing systems

This extends the results on LOGICAL DEDUCTION in §3.3.

A.2.1 Russian misconceptions.

In the MISCONCEPTIONS RUSSIAN task, the true statement must be chosen out of a pair of Russian sentences: a statement s and its negation t.

We first describe an approach that does not use translation and already performs better than random guessing – and better than baseline methods that simply select the more likely of the two statements – using the largest GPT-3 model, which has sufficient knowledge of Russian. We compute the posterior over the two hypotheses "s is true, t is false" and "s is false, t is true":

$$p_{\text{LLM}}(\text{“}\texttt{T}\text{”} \mid \text{“}\texttt{T or F? } s\texttt{. Answer: }\text{”})\, p_{\text{LLM}}(\text{“}\texttt{F}\text{”} \mid \text{“}\texttt{T or F? } t\texttt{. Answer: }\text{”}),$$

$$p_{\text{LLM}}(\text{“}\texttt{F}\text{”} \mid \text{“}\texttt{T or F? } s\texttt{. Answer: }\text{”})\, p_{\text{LLM}}(\text{“}\texttt{T}\text{”} \mid \text{“}\texttt{T or F? } t\texttt{. Answer: }\text{”}).$$

where `T` denotes `True` and `F` `False` in the actual prompt. This is a kind of **Product aggregation**. If the posterior on the first option is higher, s is chosen as the true statement; otherwise, t is chosen.

This approach can be combined with a **Translation** prompt that produces translations of s and t into English, then uses these translations in place of s and t in the above computations. The approach can be further extended by sampling a *set* of translations and performing **Mixture aggregation** over the translations. Our reported result uses 10 generated translation for each statement, but it is only 2% higher than the result using one generated translation.

A.2.2 Emoji movie

The multiple-choice EMOJI MOVIE task requires selecting the name of a movie from a list $\{m_i\}$ that is best described by a sequence of emoji symbols $s = (s_1 \ldots s_n)$. An **Order inversion** prompt performs best on this task using the Davinci variant of GPT-3: choosing the answer

$$\arg\max_i p_{\text{LLM}}(s \mid \text{“}\texttt{Emoji describing the movie } m_i\text{”}).$$

We also attempt to use a **Translation** prompt to obtain a single-word English description w_j of each emoji s_j in s, then score using

$$\arg\max_i p_{\text{LLM}}(w_1 \ldots w_n \mid \text{“}\texttt{Words describing the movie } m_i\text{”}).$$

This approach performs slightly better than **Order inversion** alone using InstructGPT. However, it does not work with the base GPT-3 models, which do not as reliably translate emoji to English.

A.2.3 Persian QA

We solve this standard extractive question answering task by simply translating the passage and question from Persian to English using a **Translation** prompt, generating English text, up to the first period or line break, following the concatenation of the translated prompt and question, and translating the result back to Persian using another **Translation** prompt.

No few-shot algorithms have above zero accuracy on this task, indicating models' knowledge is sufficient to translate between languages (probably due to the presence of paired data in the training corpus), but insufficient to reason in the source language without passing through an intermediate latent variable, the translation.

Finally, note that the accuracy is evaluated by exact string match, which contributes to the very low scores. We observed that the answers generated by **ThinkSum** are often paraphrases or terms related to the correct answers, which suggests that the result could be improved by using the knowledge that the target string always appears verbatim as a substring of the prompt.

A.3 Semantic relatedness

This extends the results on ODD ONE OUT in §3.2.

A.3.1 Phrase relatedness

Each question in the multiple-choice PHRASE RELATEDNESS task requires to determine which of a given set of words or phrases $\{w_i\}$ is related to a query phrase q. We query the LLM for the likelihood of q following a **List of words** prompt to form a vector of likelihoods:

$$p_i = p_{\text{LLM}}(q \mid \text{“}\texttt{List of words: } w_i\texttt{, }\text{”}).$$

The answer selected is the one with highest likelihood, $\arg\max_i p_i$ (a trivial **Sum** operation). We note that this is also an instance of **Order inversion**: the query is scored following a prompt in which each of the candidate answers is substituted.

A.3.2 Codenames

Each question in CODENAMES requires selecting the k words from a set $\{w_i\}$ that are most closely related to a query word q. We form a vector p_i in the same way as for PHRASE RELATEDNESS, then select the top k entries in p_i to produce the output.[2]

A.4 Substitution and aggregation

We give two other example of substitution and aggregation operations complementing the experiments on INVENTED WORDS (§3.1) and ODD ONE OUT (§3.2).

A.4.1 Novel concepts

In the multiple-choice NOVEL CONCEPTS task, a set of words or phrases $W = \{w_i\}$ and a set of statements $S = \{s_j\}$ with third-person plural pronoun subjects (‘They all...’) are given, and the statement which is true for all items in W must be determined.

We treat each statement s_j as a *template*, into which words w can be substituted by replacing ‘They all’ with w. Denoting by $s_j\langle w\rangle$ the substitution of w into s_j, we form a $|W| \times |S|$ matrix P_{ij} by scoring the **Substitution** of each word into each statement and considering the **Ratio of likelihoods** with the template without substitution: $P_{ij} = \frac{p_{\text{LLM}}(s_j\langle w_i\rangle)}{p_{\text{LLM}}(s_j)}$.We then perform **Product aggregation** to select the statement which is most likely to be generated by all words in the set. To be precise, the selected statement is $\arg\max_j \prod_i P_{ij}$.

A.4.2 Code line description

We solve the CODE LINE DESCRIPTION task, in which a correct comment for a code snippet is to be chosen, using **Order inversion** and **Substitution** techniques.

The greatest gain – amounting for all but 1% of the improvement relative to direct prompting – arises from **Order inversion**. Instead of ranking the candidate comments c by their likelihood following the given code s (i.e., $p(c|s)$), we score each candidate comment c by the likelihood of the code to follow c formatted as a Python comment ($p(s|\text{“}\#\ c\text{”})$).

We also experimented with **Substitution** and **Product aggregation**, which yielded an additional small accuracy gain. The code snippets are written in Python, which requires code to be formatted using an arbitrary but consistent number of spaces for line indentation. Using the knowledge that the correct comment should be most likely to generate the program in *any* of its equivalent representations, we scored comments in the manner described in the preceding paragraph, but with s reformatted with different number of indentation spaces n. The resulting scores were then multiplied over $n = 1, 2, \ldots, 6$ and the highest-scoring comment selected.

[2] Because the task is evaluated by BLEU score against the reference answers listed in alphabetical order, we perform the additional step of converting the top indices to the answer in the right format. Alphabetization of short lists is trivial in code, but can also very reliably be done by prompting GPT-3.

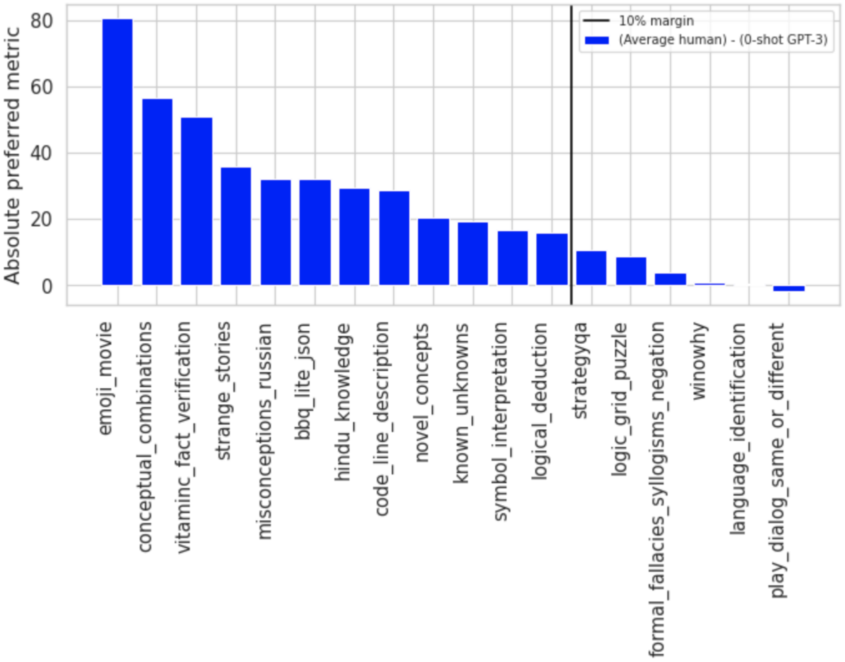


Figure B.1: Margin between 0-shot GPT-3 and average human performance for BIG-bench Lite tasks. Using **ThinkSum**, we address many of the tasks that have greater than 10% performance margin with average human, and significantly reduce and often overturn the margin.

A.5 Other tasks

A.5.1 Language identification

The multiple choice LANGUAGE IDENTIFICATION task is similar in form and solution to CODE LINE DESCRIPTION and we include it for completeness to show the large difference that can be made by **Order inversion**.

Rather than scoring all candidate language names ℓ following the given sentence s (i.e., $p(s|\ell)$), we instead score each language name ℓ by $p(s|$"`The following is a sentence in` ℓ`:`") and select the highest-scoring ℓ as the answer.

B BIG-bench Lite

Figure B.1 shows the performance margin between an average human and zero-shot GPT-3 on tasks in BIG-bench Lite, a select subset of tasks chosen by the authors of the benchmark to showcase the most important aspects of LLMs that need improvement. The vertical black bar separates the dataset into tasks where GPT-3 is already within the margin of just 10% compared to the average human accuracy, and the harder tasks (on the left). We show in the main text that some of these harder tasks, in particular EMOJI MOVIE, CONCEPTUAL COMBINATIONS,KNOWN UNKNOWNS, NOVEL CONCEPTS, MISCONCEPTIONS RUSSIAN and LOGICAL DEDUCTION, the margins are shrunk considerably, often exceeding average human performance. Other tasks in BIG-bench lite such as LOGIC GRID PUZZLE and SYMBOL INTERPRETATION share a similar structure to the addressed by **ThinkSum**, and thus could be investigated as part of future work. Another example where **ThinkSum** can be applied is the CODE LINE DESCRIPTION task, where we observe in our preliminary experiments that a simple order inversion can significantly outperform average human accuracy.

C Task descriptions

C.1 Hallucination detection

Known unknowns. In this task, the aim is to measure the ability of LLMs to identify whether the answer to a question is known, or unknown. If the answer to the question cannot be known, an imaginary answer is generated, and the model is asked to choose between the imaginary answer and the word “Unknown”. If the question has a correct answer, the correct answer and “Unknown" are provided as options. An example question from this task is *“How often did Abraham Lincoln cut his toenails?”*, and the options are *“Every Saturday night”, “Unknown”*. Similar to the sports understanding task, we can use an LLM to generate other plausible answers with the following prompt: *How often did Abraham Lincoln cut his toenails? A possible answer is: \n 1. Every Saturday night. List 4 other possible answers in the same format as the first: \n 2.* Then, the answer given in the question is predicted to be “known” if its posterior is higher by the second most likely option by some margin. In our experiments, we chose this value to be $1/N_e$ where N_e is the number of examples, including the original option.

Sports understanding. In this task, artificial sentences are constructed using the name of professional sports players and actions from particular sports. The model is then asked to identify whether the sentence is plausible, where a sentence is considered plausible if the sport of the player matches the sport of the action described in the sentence. An example from the task is *“Statement: Draymond Green threw a touchdown. Plausible/implausible?”*

For **ThinkSum** to be able to parse outputs, GPT-3 generations need to be in a pre-determined format. While larger models can obey a format without demonstrations, for smaller models it is helpful to demonstrate the format with an example. Thus, we use the following prompt: “*List 4 examples of players who scored a rabona goal.\n 1. Cristiano Ronaldo\n 2. Erik Lamela\n 3. Mario Balotelli\n 4. Angel Di Maria\n List 4 examples of players who threw a touchdown.\n1.*”.

C.2 Translation

Emoji movie. In this task, the LLM is presented with emojis that describe the plot of a movie, and the model is asked to guess the correct movie from the given options. The task evaluates the LLMs understanding of emojis, and its knowledge of popular movies. An example from the task is *“What movie does this emoji describe? {lying-face-emoji} \n, pinocchio \n heat \n the big sleep, \n inception \n lilo & stitch*

Misconceptions Russian. In this task, the LLM is presented with sentences in Russian that contain misconceptions, and their true counterparts. The task evaluates if the LLM believes common misconceptions. One example of a target pair from the task, translated to English is as follows: *“Vaccines do not cause autism. \n Vaccines cause autism.”*

ParsiNLU reading comprehension. This is a standard extractive QA task in which a short passage and question are provided in Persian and the model must generate the answer, also in Persian.

C.3 Semantic relatedness

Phrase relatedness. In this task, an input phrase, and four target phrases are presented to the language model. The language model is asked to identify the most related choice from the listed target options. An example from the task is *“For each word or phrase, identify the most related choice from the listed options. \n Input: home town \n Option: town center \n Option: location \n Option: native city \n Option: home run”*

Codenames. In this task, the language model is asked to identify words associated with a given word. An example from the task is *“Try to identify the 2 words best associated with the word WHITE from the following list: \n book, anchor, rainbow, shoulder, tunnel, sack, drum, pacific, page, mark, gear, glacier. Give your answer in alphabetical order.”*

Odd one out. This task is aimed at evaluating the capability of LLMs in semantic relatedness. This task presents the model with four to six words, where all words except one word are semantically or grammatically related to each other. The goal for the language model is to identify the odd word. An example question from the task is *“Pick the odd word out: glass, head, arm, leg, hand, foot”*.

C.4 Concept understanding

In the following tasks, the shared goal is to test the ability of LLMs on concepts over entities that have likely not been observed during training.

Conceptual combinations: Invented words. In this task, the LLM is provided with two invented words, and their definitions in the input. The LLM is then asked to infer the most plausible meaning resulting from the combination of the invented words. As the words are invented, they are not present in the training set, and the LLM needs to understand and combine the definitions of the invented words to reason about the meaning of the combination. An example is: *"The word 'binne' means any animal that is furry and has four legs, and the word 'bam' means a simple sort of dwelling. Question: Which of the following sentences best characterizes binne bams?"*. Similar to SPORTS UNDERSTANDING, we can use the following prompt to force the LLM to obey a fixed format: *"List synonyms of binne, separate synonyms by comma:"*

Novel concepts. In this task, the LLM is presented with two to four disparate entities that typically would not co-occur frequently, but share an underlying conceptual or linguistic concept. The aim is to test the ability of the LLM to reason about entities that are unlikely to have been observed in the same context during training. In a multiple-choice setting, the LLM is given concepts relating to the entities, and is asked to generate the intended concepts against carefully chosen tempting distractors. The choices are not presented in the prompt. An example question from the task is as follows: *"What do the following have in common? 1) bumble bees 2) 01010101 3) race cars"*, and the answer options are *They all make noise, "They all are yellow, They all are binary, They all go fast, They all have stripes"*.

C.5 Other tasks

Two multiple-choice tasks test the LLM's knowledge of specific domains, such as uncommon languages and programs.

Code line description. This task requires the LLM to select the appropriate text description, out of four choices, for a short snippet of Python code, that could act as a comment describing the behaviour of a function.

C.5.1 Language identification.

This task requires the LLM to select, out of eleven choices, the language in which a text is written. The languages represent a diversity of language families and writing systems and most are very infrequent in text found on the Internet.

D Additional experimental details

Our experiments are performed using four different sizes of GPT-2 (Small, Medium, Large, and XL) (Radford et al., 2019), GPT-3 with four different model sizes (ada,babbage,curie,davinci) (Brown et al., 2020), and InstructGPT (Ouyang et al., 2022). All GPT-3 experiments are run between August 2022 and September 2022 by using the OpenAI API. Our GPT-2 experiments were run in PyTorch (Paszke et al., 2019) and the Hugging Face Transformers library with a Tesla K80 GPU.

D.1 Hyperparameters

Maximum generation length. For tasks that require **example and list generation**, such as CONCEPTUAL COMBINATIONS, KNOWN UNKNOWNS, and SPORTS UNDERSTANDING, we use max_tokens = 100. For **fact generation** in ODD ONE OUT with auxiliary knowledge and **ThinkSum**, we use max_tokens = 1000.

Temperature. All GPT-2 experiments used temperature = 0.5. For SPORTS UNDERSTANDING and translation tasks, we used temperature = 0.5 to promote diversity of generated plausible options. All other experiments used temperature = 0 (greedy decoding).

Number of examples (N_e). For CONCEPTUAL COMBINATIONS we used $N_e = 2$, and for KNOWN UNKNOWNS and SPORTS UNDERSTANDING we used $N_e = 4$.

Threshold. A threshold of 0.01 was used for SPORTS UNDERSTANDING.

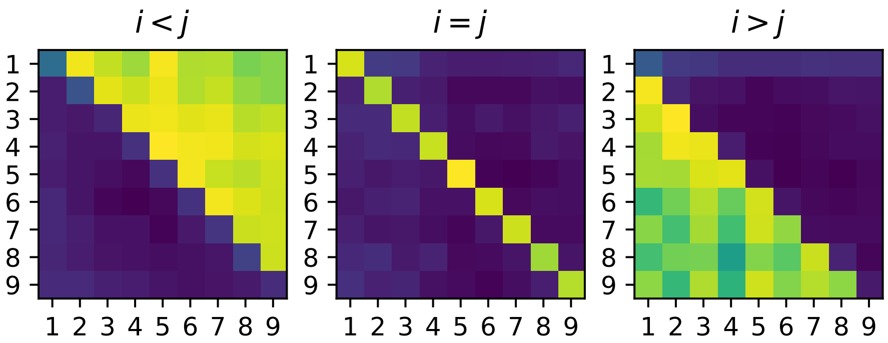
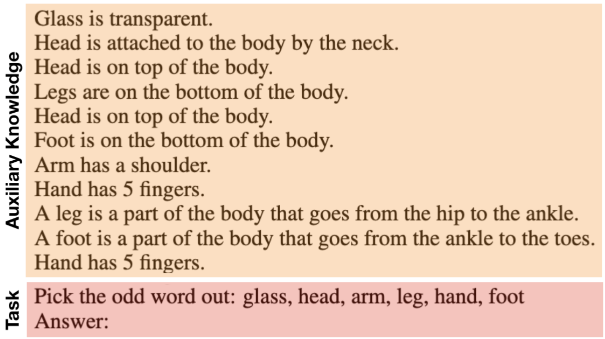


Figure D.1: Probabilities of different (in)equalities according to GPT-3 text-davinci-002 (logit).


Auxiliary Knowledge
Task

Glass is transparent.
Head is attached to the body by the neck.
Head is on top of the body.
Legs are on the bottom of the body.
Head is on top of the body.
Foot is on the bottom of the body.
Arm has a shoulder.
Hand has 5 fingers.
A leg is a part of the body that goes from the hip to the ankle.
A foot is a part of the body that goes from the ankle to the toes.
Hand has 5 fingers.

Pick the odd word out: glass, head, arm, leg, hand, foot
Answer:

Figure D.2: Auxiliary knowledge prompting applied to ODD ONE OUT. Facts are generated using the "list differences" prompt described in Figure 2 (right) and post-processed according to §D.3.

D.2 Using an LLM to evaluate inequalities.

Using GPT-3 or external algorithms to evaluate inequalities. We show how a LLM can be used to find the truth values of inequalities involving small numbers, rather than resorting to calls to an external system that is aware of arithmetic. Fig. D.1 shows the matrix of posterior probabilities evaluated using InstructGPT (text-davinci-002) for strings of form "x=y", "x<y", "x>y" for $x, y \in \{1, .., 9\}$. The probabilities are computed using prompts of the form "`True or false:` x<y`? The answer is:`" and normalizing the probability of the first token over the two options "`true`" and "`false`". These are the probabilities evaluated in (1).

D.3 Knowledge generation details

Post-processing. In our knowledge generation experiments for both **ThinkSum** and the auxiliary knowledge approach, we post-process the generated knowledge statements, to ensure formatting does not harm the predictions of each method. We first remove the extra spaces and the numbers and punctuation generated by the LLM before each fact while enumerating the items of the list. Later, we only keep sentences that contain only one of the objects of interest from the task, to make sure each sentence contains a knowledge statement into which any of the objects can be substituted. Finally, sentences with less than 3 words are removed as these are not likely to contain informative statements.

Auxiliary knowledge. For auxiliary knowledge experiments, we prepend the generated and post-processed knowledge statements before the question in the task. An example is illustrated in Figure D.2.

D.4 Inference Cost for ThinkSum

The inference cost for ThinkSum scales with the number of parallel calls to the LLM, which is determined for each task by the number of **Think** prompts used and the number of objects for which likelihood computations are required at the **Sum** stage. For the tasks that we considered, as the number of **Think** prompts is not typically high and the prompts are short, the inference cost increase is marginal. In some cases, **ThinkSum** is faster than chain-of-thought prompting due to its ability to perform parallel calls to the LLM. For instance, **ThinkSum** is 23% faster for PHRASE RELATEDNESS compared to chain-of-thought approaches with 5 facts generated using InstructGPT.

E Expectation Maximization

We model items $i \in I$ and facts $f \in F$ as being generated from a latent class $c \in \{0, 1\}$. The distribution is modeled as:

$$P(i, f \mid c) = P(i \mid c)P(f \mid c) \quad P(i, f) = \sum_c P(c)P(i, f \mid c)$$

where $P(i, f)$ is a matrix of likelihoods from the LLM and the semantic components, groupings $P(i \mid c)$ and $P(f \mid c)$. The iterative expectation-maximization (EM; Dempster et al., 1977) algorithm to derive $P(i \mid c)$ and $P(f \mid c)$ has the following updates:

$$\begin{aligned}
Q(c \mid i, f) &\propto P(i \mid c)P(f \mid c)P(c) \\
P(i \mid c) &\propto \sum_f P(i, f)Q(c \mid i, f) \\
P(f \mid c) &\propto \sum_i P(i, f)Q(c \mid i, f) \\
P(c) &\propto \sum_{i,f} P(i, f)Q(c \mid i, f)
\end{aligned}$$

where $Q(c \mid i, f)$ is the posterior distribution over the latent class c that we maintain for each pair (i, f). EM is run for 200 iterations, which is more than sufficient for convergence.

Odd one out	Words: blue, pink, magenta, banana All words are colors except banana. The odd one out is banana. Words: pencil, eraser, baby, rule, notebook All words are office supplies except baby. The odd one out is baby.
Phrase relatedness	For each word or phrase, identify the most related choice from the listed options. Input: Ice Cream Option: Antarctica Option: Titanic Option: Dessert Option: Sour Cream Ice cream is a type of dessert. Therefore, ice cream and dessert are the most related. Answer: Dessert
Known unknowns	What was the population of San Francisco in 2018? Option: 879,676 Option: Unknown The question asks the population of San Francisco in 2018, for which data can be collected. Population data for cities on a yearly basis is available, and thus the answer is known, and it is 879,676. Answer: 879,676 What was the population of San Francisco yesterday? Option: 891,402 Option: Unknown The question asks the population of San Francisco yesterday. As it is not possible to know the exact population of a city on a daily basis, the answer for this question is unknown. Answer: Unknown
Logical deduction	On a table, there are five plates: a black plate, a white plate, a green plate, a blue plate, and a red plate. The white plate is bigger than the green plate. The red plate is the biggest. The black plate is bigger than the blue plate. The black plate is smaller than the green plate. Which plate is the smallest? Option: The red plate is the smallest. Option: The black plate is the smallest. Option: The white plate is the smallest. Option: The green plate is the smallest. Option: The blue plate is the smallest. The black plate is bigger than the blue plate. The black plate is smaller than the green plate, as a result the green plate is bigger than the blue plate as well. The white plate is bigger than the green plate, which is bigger than the blue plate. As a result, the green plate is bigger than the blue plate. The red plate is the biggest, so it is bigger than the blue plate. Since all other plates are bigger than the blue plate, the blue plate is smallest. Answer: The blue plate is the smallest.
Invented words	The word ’borger’ are animals who bite specific things for fun, and the word ’folpt’ is a type of a chewy toy. Question: Which of the following sentences best characterizes borger folpts? Option: Borger folpts are leashes for animals. Option: Borger folpts are toys for infants. Option: Borger folpts are hard to swallow. Option: Borger folpts are pet toys. Borgers are animals, and folpts are chewy toys. Therefore, borger folpts are chewy toys that animals, or pets, can play with. Therefore, the answer is borger folpts are pet toys. Answer: Borger folpts are pet toys.

Table A.1: Few-shot demonstrations used for chain of thought (Table 4).

Task: Phrase relatedness **Input:** For each word or phrase, identify the most related choice from the listed options. Input: home town Option: town center Option: location Option: native city Option: home run
Task: Odd one out **Input:** Pick the odd word out: glass, head, arm, leg, hand, foot
Task: Conceptual combinations: Invented words **Input:** The word ‘binne’ means any animal that is furry and has four legs, and the word ‘bam’ means a simple sort of dwelling. Question: Which of the following sentences best characterizes binne bams? Option: Binne bams are for pets. Option: Binne bams are typically furnished with sofas and televisions. Option: Binne bams are luxurious apartments. Option: Binne bams are places where people live
Task: Novel concepts **Input:** What do the following have in common? 1) bumble bees 2) 01010101 3) race cars Option: They all make noise. Option: They all are yellow. Option: They all are binary. Option: They all go fast. Option: They all have stripes.
Task: Logical deduction: Five objects **Input:** On a shelf, there are five books: a gray book, a red book, a purple book, a blue book, and a black book. The red book is to the right of the gray book. The black book is to the left of the blue book. The blue book is to the left of the gray book. The purple book is the second from the right. Option: The {gray, red, purple, blue, black} book is second from the right.
Task: Codenames **Input:** Try to identify the 2 words best associated with the word WHITE from the following list: book, anchor, rainbow, shoulder, tunnel, sack, drum, pacific, page, mark, gear, glacier. Give your answer in alphabetical order.
Task: Known unknowns **Input:** How often did Abraham Lincoln cut his toenails? Option: Every Saturday night Option: Unknown
Task: Sports understanding **Input:** Statement: Draymond Green threw a touchdown. Plausible/implausible?
Task: Emoji movie **Input:** What movie does this emoji describe? {lying-face-emoji} Option: pinocchio Option: heat Option: the big sleep Option: inception Option: lilo & stitch
Task: Misconceptions Russian **Input:** Vaccines cause autism. / Vaccines do not cause autism. *[in Russian]*

Task: Code line description
Input:
`for i in range(23):`
`  print(i)`
Option: prints values from 0 to 22,
Option: computes first 10 prime numbers,
Option: prints values from 1 to 10,
Option: prints ’hello world’ to the terminal

Task: ParsiNLU reading comprehension **Input:** To reduce fever, use over-the-counter medications such as acetaminophen and ibuprofen. Note the appropriate dosage and do not use them alongside other fever-reducing medications. You should not give aspirin to your baby without consulting a doctor. Babies under 6 months of age should not be given ibuprofen. What brings down fever? *[in Persian]*
Task: Language identification **Input:** Given a sentence, select the correct language among the choices. Mi texaas o a mu vipin simi ri xavil ina vipin si Krais xa. E mi lamon o ne taa siak a xavil ina vipin si Krais e faxuvule xuvul pana vipin sina tefin aava lisan xolane, piau paaliu! Options: Assamese, Nandi, Patamona, Chavacano, Kapingamarangi, Turkish, Kara, Bribri, Gofa, Pali, Shatt

Table D.1: List of examples from the studied BIG-bench tasks.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
See "limitations" section on p.9.

☒ A2. Did you discuss any potential risks of your work?
We see no risks beyond those already inherent in large language models, but we include Limitations and Ethics sections before the references (p.9).

☑ A3. Do the abstract and introduction summarize the paper's main claims?
See the abstract and introduction.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

We use existing models and datasets. See following answers.

☑ B1. Did you cite the creators of artifacts you used?
See the introduction, where we cite the BIG-bench suite.

□ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Not applicable. Note that the BIG-bench benchmark, which we use, is licensed for use in academic work such as ours.

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We use the BIG-bench suite. In the introduction, we describe it and summarize its motivations.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We used an existing large-scale benchmark to evaluate pretrained language models. We believe the data for the specific tasks we studied is very unlikely to contain such content, which should be clear from the task examples (last page of the paper), although this may not be true of all tasks in the BIG-bench suite.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
See the task descriptions in Appendix D.

☒ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
We used existing benchmarks (BIG-bench) for which extensive documentation exists.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

See section 3 and the Appendix.

☒ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
We use the OpenAI API to run experiments with GPT-3-family models, which accounts for the bulk of the computational cost. However, the exact cost is unknown. On the order of 250k queries were made to the API to obtain the results in the paper.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
See Appendix E.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Most of the experiments are deterministic. A few experiments use sampled decoding of large language models (at low temperature), and we describe the settings in Appendix E.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
See Appendix E.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

□ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

□ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

□ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

□ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

□ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.