THINKSUM: PROBABILISTIC REASONING OVER SETS USING LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

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 largest LLMs fail in scenarios that require reasoning over multiple objects or facts or making sequences of logical deductions. We propose a two-stage probabilistic inference paradigm, THINKSUM, that reasons over sets of objects or facts in a structured manner. In the first stage (THINK – 'fast' 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 – 'slow' probabilistic inference or reasoning), the results of these queries are aggregated to make the final prediction. We demonstrate the advantages of THINKSUM on the BIG-bench suite of evaluation tasks, achieving improvements over the state of the art using GPT-family models on ten difficult tasks, often with far smaller model variants. We compare and contrast THINKSUM with other proposed modifications to direct prompting of LLMs, such as variants of chain-of-thought prompting. We argue 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.

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 that are formulated as generation or selection of a likely completion of a prompt. This formulation requires LLMs to perform associative **fast 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. Fast thinking is made possible by the compression in the LLM weights of information that is repeated in a variety of ways in large training datasets.

However, it is increasingly evident that when **reasoning**, or **slow thinking**, is required, failure modes of LLMs are revealed. In our usage, reasoning is 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. 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 though definitions (CONCEPTUAL COMBINATIONS), etc. These are tasks where a human solver's intuitive feeling of '(in)coherence' is not sufficient to produce the right answer: the solution is obtained by a sequence of thoughts that can be explained in words and may even require writing down intermediate results if working memory is insufficient.

We show on several examples in BIG-bench that such problems can be addressed by adding a simple two-component mechanism, which we name THINKSUM:

- THINK (fast thinking / association / knowledge retrieval step): creating an association of spans of text 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', . . . }

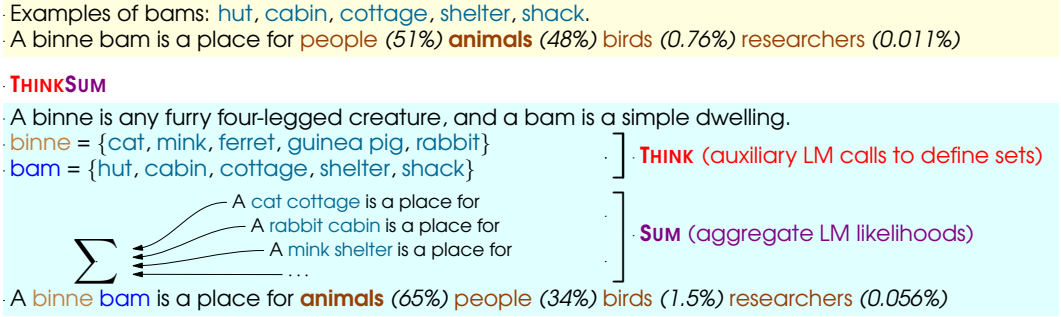

Figure 1: An example adapted from the CONCEPTUAL COMBINATIONS (INVENTED WORDS) task, in which models must select the most likely completion of a phrase that includes nonce words whose definitions are given. **Direct prompting** evaluates completion likelihoods normalized over the four answer choices ('people', 'animals', 'birds', 'researchers'). **Chain-of-thought**-like or **auxiliary knowledge** approaches would query a LLM or knowledge base for additional context. Our **THINKSUM** approach to this task queries a LLM to produce sets of examples defining the nonce words, then marginalizes over substitutions of these examples into the target phrase. The LLM is GPT-2 XL.

by prompting GPT-3 with the definition and asking for examples. However, it may also consist of scoring alone, in order to form a matrix of probabilities over 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. The 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.

THINKSUM is named by analogy with other algorithms with two basic operations that 'expand' and 'aggregate', like MapReduce in distributed computing and sum-product in probabilistic graphical models.

We discuss different ways to **THINK** and to **SUM** in section §2, but we start with one example, illustrated in Fig. 1, motivated by the INVENTED WORDS subtask of the CONCEPTUAL COMBINATIONS task in BIG-bench (Srivastava et al., 2022). 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 sentence that uses a 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. 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. In this way, by mapping individual spans of the text of interest to sets we arrive at a mixture model (in this example with 25 components, for 5 possible replacements of each of the words), which can be used in the same manner the original LLM is used, to either score text, or to generate it token by token. In this case, 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 most likely.

An important difference between **THINKSUM** and existing chain-of-thought-like prompt engineering methods (Wei et al., 2022; Kojima et al., 2022), is that the reasoning step is not reduced to a generation problem for the LLM, but is perfomed as a probabilistic inference external to the LLM. This reduces its vulnerability to features of the prompt, such as accidental distraction of the LLM by spurious patterns. 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. Fig. 1 shows the potential brittleness of such 'reasoning', especially in smaller models, which have stronger recency bias (Malkin et al., 2022): if we simply list generated examples as additional context in the prompt, the recency bias causes the LLM to still give higher probability to 'people' than to 'animals', simply because 'bam' – simple dwelling – examples are given after the 'binne' examples.

Imposing an alternative reasoning system over an associative "knee-jerk reaction" system has an analogy with models of human cognitive processes (Tversky & 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, and, in the context of reasoning with deep learning models, has been interpreted as operating with sparse concepts that can be described in language (Bengio, 2017; Bengio & Goyal, 2020). Through repeated usage, the slow-thinking functions of System 2 can become efficiently 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 System 2 controller that allows for probabilistic inference.

2 FAST AND SLOW THINKING WITH LLMS

The example in Fig. 1 falls into a general **THINKSUM** paradigm which extends the reasoning capabilities with a given model by explicitly associating certain *text spans* with *sets of other strings*, which may serve as alternatives or elaborations/explanations and can be either defined by the user or inferred by the LLM itself. These associations then provide multiple texts to be evaluated, again by the LLM itself. The collection of resulting probabilities provides an opportunity to summarize the text using standard probabilistic inference techniques, which usually include a summation.

2.1 HOW TO THINK

Here we list examples of the "fast thinking" that precedes the summarization stage, referring to some tasks from the BIG-bench lite suite.

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 that can be seen as being made by a controller. For multiple-choice questions, the simplest way is to write it as a Cloze task. 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 operations that are reliably achieved by standard NLP tools, such as ensuring subject-verb agreement after the substitution.

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 'furry four-legged creature', into a list of examples. This can be achieved with a prompt such as "`A binne is a furry four-legged creature. Examples: 1.`". The generated completion can then 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 THINKing 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, in place of the concept about which facts are generated. A variation on this asks the LLM to generate differences between two concepts, as shown in Fig. 4.

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 a similar meaning.
- 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).

- **Posterior computation** by normalizing probabilities over a set.
- **Majority/minority vote (argmin/argmax)**: selecting of the most likely answer is a component of nearly any answer selection procedure.
- **Thresholding**: An answer depends on the value of a probability (or a difference or ratio of probabilities). This can be used when a discrete answer needs to be produced from a set of output likelihoods.
- **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 partial prompt 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

We compare THINKSUM with the published results for n-shot GPT-3 175B (davinci) in BIG-bench, where $n \in \{0, 1, 2, 3\}$. Our main results are shown in Table 1. Below, we describe THINKSUM for each task. Detailed descriptions are in Appendix §B, and examples of each task appear in Table D.1.

3.1 Semantic relatedness

3.1.1 Introductory examples: Phrase relatedness and Codenames

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** THINK prompt to form a vector of

Table 1: Accuracy for GPT-3 175B (davinci), and **THINKSUM** with 175B (davinci), InstructGPT and GPT-2 XL on ten BIG-bench tasks.

		GPT-3 (davinci) n-shot				**THINKSUM**		
Task	Avg. H	$n = 0$	1	2	3	GPT-3	InstructGPT	GPT-2 XL
PHRASE RELATEDNESS (§3.1.1)	0.74	0.37	0.42	0.52	0.59	0.85	**0.87**	0.79
CODENAMES (§3.1.1)	0.18	0.01	0.11	0.16	0.19	0.37	**0.41**	0.36
ODD ONE OUT (§3.1.2)	0.80	0.27	0.20	0.23	0.23	0.80	**0.84**	0.71
NOVEL CONCEPTS (§3.2.1)	0.67	0.47	0.47	0.56	0.56	0.72	**0.75**	0.50
INVENTED WORDS (§3.2.2)	N/A	0.29	0.14	0.14	0.21	0.64	**0.71**	0.29
SPORTS UNDERSTANDING (§3.3.1)	0.71	0.50	0.50	0.50	0.50	0.71	**0.74**	0.54
KNOWN UNKNOWNS (§3.3.2)	**0.80**	0.61	0.52	0.48	0.50	0.54	0.76	–
MISCONCEPTIONS RUSSIAN (§3.4.1)	0.65	0.33	0.33	0.41	0.35	**0.70**	0.61	–
EMOJI MOVIE (§3.4.1)	**0.93**	0.12	0.18	0.12	0.19	0.80	0.75	–
FIVE OBJECTS (§3.4.2)	N/A	0.23	0.29	0.28	0.32	–	**0.77**	–

likelihoods:

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

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.

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.[1]

3.1.2 ODD ONE OUT

We examine possible **THINK** and **SUM** approaches in greater depth on the multiple-choice ODD ONE OUT task, in which the word in a set $W = \{w_i\}$ that is least semantically related to the others must be chosen.

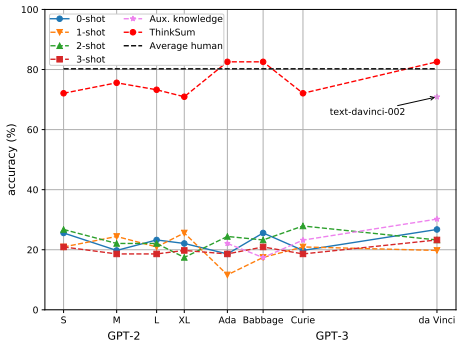

Figure 2: Performance of GPT-3 (n-shot, $n = 0, 1, 2, 3$), auxiliary knowledge, and **THINKSUM** on ODD ONE OUT with various model sizes.

List of words. We first consider an approach similar to that used in §3.1.1. We form a matrix P_{ij} using a **List of words THINK** prompt for each pair of indices i, j:

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

This matrix is aggregated by averaging over j and selecting the i with lowest average, i.e., least likelihood of being generated by a mixture of all words in the set: $i = \arg\min_i \frac{1}{|W|} \sum_j P_{ij}$. This is a case of the **Mixture aggregation** operation.

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).

Fact generation. As an alternative approach, we use a **Fact generation** prompt. An effective way to mine facts for semantic relatedness tasks is two consider two items in the same context, in order to get relevant facts regarding how items are related to each other. One prompt to achieve this is to ask for differences of pairs in the question, illustrated in Fig. 4. 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 **Fact generation** prompt, we obtain a collection of statements $S = \{s_i\}$ about items w_j We treat each s_i as a template into which different words w

[1] 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.

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 $P_i j$, defined by

$$P_{ij} = p_{\text{LLM}}(s_i\langle w_j\rangle)$$

Then, we can perform minority voting, and 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. Followed by the recent success of prompting techniques, 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 is provided in Appendix D.2 In Figure 3, we show that the accuracy of **Fact generation** based **ThinkSum** rises as the number of used generated facts is increased, while the auxiliary knowledge technique peaks and then degrades.

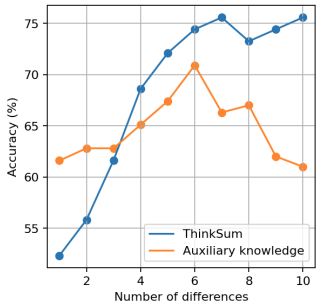

Figure 3: Performance of auxiliary knowledge vs **ThinkSum** on Odd one out with varying number of differences.

Fig. 2 how performance varies with the size of the LLM model used for GPT-3, auxiliary knowledge and **ThinkSum**, on Odd one out. Even with small GPT-2, **ThinkSum** dramatically improves over much larger largest zero- or few-shot models with or without auxiliary knowledge. The latest 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**, providing some experimental evidence to our claim that while new models may create qualitative jumps, **ThinkSum** can push the performance limits of smaller model variants through slow thinking. Additional experiments on auxiliary knowledge are provided in Appendix C.

3.2 Substitution and aggregation

3.2.1 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 $\arg\max_j \prod_i P_{ij}$.

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 4: Prompt used to generate knowledge statements for the task Odd one out.

3.2.2 Invented words

In the multiple-choice Invented words task, a pair of nonce words x_1, x_2 is defined and the correct statement must be chosen out of a set of statements $S = \{s_j\}$ that begin with (possibly an inflected form of) "x_1 x_2" (see Fig. 1).

We use an **Example generation** prompt to obtain a set of example words fitting the definition 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:

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

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

3.3 Uncertainty and Hallucination Detection

LMs 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).

3.3.1 Sports Understanding

Questions in the Sports Understanding task 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 is the sport with which player x is associated coincides with the sport in which action a is performed. We consider a generic 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 of 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. 5.

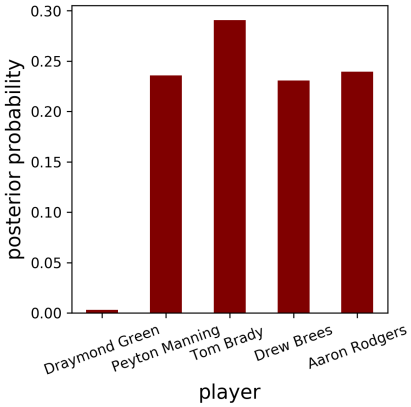

Figure 5: Example posterior probabilities generated from text-davinci-002 for sports understanding with the description *"threw a touchdown"*. The basketball player given in the question *Draymond Green* has a much lower posterior probability than the generated football players, from which we conclude the sentence *"Draymond Green threw a touchdown."* is implausible.

3.3.2 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 q, similar to that used for Sports Understanding:

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

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

3.4 Translation

3.4.1 Translating between languages and writing systems

Russian misconceptions. In the Misconceptions Russian task, the true statement must be chosen out of a pair consisting of a statement s and its negation t, with both statements written in Russian.

(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."

(b) Think Prompt	(c) LLM Output
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:	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

Figure 6: Details for LOGICAL DEDUCTION. (a) Example question from the task, (b) **THINK** prompt (c) example LLM output.

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 two choices, corresponding to the hypotheses "s is true, t is false" and "s is false, t is true":

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

and

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

where `T` denotes `True`, and `F` denotes `False` in the actual prompt. This can be seen as 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 places 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.

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 $s = (s_1 \dots s_n)$. An **Order inversion** prompt performs best on this task using the Davinci variant of GPT-3: choosing the answer

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

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

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

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

3.4.2 LOGICAL DEDUCTION

In the LOGICAL DEDUCTION task, different types of items, and clues regarding their placement are provided, as shown in Fig. 6(a). The goal is to assign probabilities to assignments of objects to their placements. The task requires parsing information about multiple objects and their relationships, understanding rules regarding ordered objects in various scenarios, and iterative application of 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 iteratively apply these rules. Here, we can

use a **Translation** prompt to map the given problem into a set of mathematical (in)equalities (Fig. 6(b)).

A generic **Translation** prompt as in Fig. 6(b) containing generic ordering statements and object names that are not used in the task is sufficient to perform the translation from natural language to mathematical equations, as shown in Fig. 6 (c). To solve the problem, we replace the problem statement with its translation T consisting of (in)equalities, and map each of the objects to the set of strings corresponding to numbers from 1 to N, where N is the number of objects. The probabilities for the target statements can now be obtained by:

$$p(\textit{yellow book=2}|T) \propto \sum_{B \in [1..N]^N} p_{\text{LLM}}(T, \textit{yellow book=2}) \tag{1}$$

where *yellow book=2* is the target option to be evaluated, B denotes the set of books, and T is obtained from the **Translation** prompt as a set of string. In the problem listed above, we get $T = \{\textit{yellow book} > \textit{purple book}, \textit{black book} < \textit{purple book}, \textit{black book=3}, \textit{brown book=1}\}$, and $p(T) = \prod_t p(T_t)$, where T_t are individual strings with items replaced by numbers. After getting T, the summation over possible orderings can be performed by evaluating sets of (in)equalities by an external algorithm. Another possibility is to use the understanding of (in)equalities in GPT-3. Details regarding this approach are provided in Appendix §C.

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., 2022; Creswell et al., 2022; Wang et al., 2022a; Liu et al., 2022b). 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 & Pavlick (2022) observed that instructive prompts perform similarly with misleading or intentionally irrelevant prompts. Additionally, Ye & 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.

Probabilistic inference. More recent approaches have proposed probabilistic inference approaches for tackling true/false question answering and commonsense question answering (Jung et al., 2022; Liu et al., 2022a). Dohan et al. (2022) formalizes existing prompting techniques and presents a unified probabilistic perspective to prompting. 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 the prompt itself, which prevents 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.

REFERENCES

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

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

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. *Neural Information Processing Systems (NeurIPS)*, 2020.

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.

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

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

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.

Nouha Dziri, Andrea Madotto, Osmar Zaïane, and Avishek Joey Bose. Neural path hunter: Reducing hallucination in dialogue systems via path grounding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2197–2214, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.168. URL https://aclanthology.org/2021.emnlp-main.168.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 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*, pp. 7038–7051, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.564. URL https://aclanthology.org/2021.emnlp-main.564.

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.

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

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

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. Generated knowledge prompting for commonsense reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3154–3169, Dublin, Ireland, May 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.225. URL https://aclanthology.org/2022.acl-long.225.

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

Zihan Liu, Mostofa Patwary, Ryan Prenger, Shrimai Prabhumoye, Wei Ping, Mohammad Shoeybi, and Bryan Catanzaro. Multi-stage prompting for knowledgeable dialogue generation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 1317–1337, Dublin, Ireland, May 2022b. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.104. URL https://aclanthology.org/2022.findings-acl.104.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 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)*, pp. 8086–8098, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.556. URL https://aclanthology.org/2022.acl-long.556.

Nikolay Malkin, Zhen Wang, and Nebojsa Jojic. 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)*, pp. 8214–8236, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.565. URL https://aclanthology.org/2022.acl-long.565.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 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)*, pp. 5316–5330, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.365. URL https://aclanthology.org/2022.acl-long.365.

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

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

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

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

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

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

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 3784–3803, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.320. URL https://aclanthology.org/2021.findings-emnlp.320.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4615–4629, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.373. URL https://aclanthology.org/2020.emnlp-main.373.

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. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

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

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

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*, 2022b.

Albert Webson and Ellie Pavlick. 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*, pp. 2300–2344, Seattle, United States, July 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.naacl-main.167.

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 preprint arXiv:2201.11903*, 2022.

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

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

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

Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Francisco Guzmán, Luke Zettlemoyer, and Marjan Ghazvininejad. Detecting hallucinated content in conditional neural sequence generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 1393–1404, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.120. URL https://aclanthology.org/2021.findings-acl.120.

Denny Zhou, Nathanael Schärli, 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 preprint arXiv:2205.10625*, 2022.

A Appendix

A.1 BIG-bench Lite

Figure A.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% comapred 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.

B Additional task descriptions

B.1 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

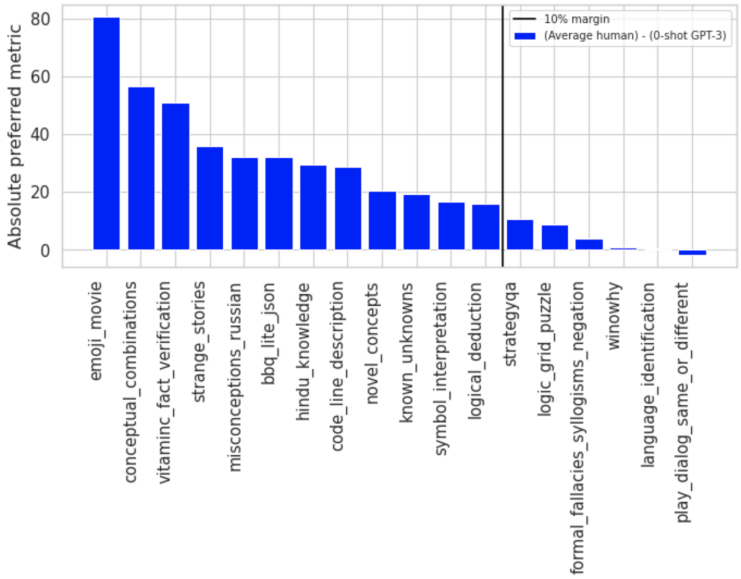

Figure A.1: Margin between 0-shot GPT-3 and average human performance for BIG-bench Lite tasks. Using **THINKSUM**, we address many of the tasks that has greater than 10% performance margin with average human, and significantly reduce, and often overturn the margin.

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"*.

B.2 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 the sports understanding 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

Table B.1: **THINKSUM** vs. auxiliary knowledge

	ODD ONE OUT	PHRASE RELATEDNESS	LOGICAL DEDUCTION
THINKSUM	0.84	0.87	0.77
Auxiliary knowledge	0.71	0.75	0.50

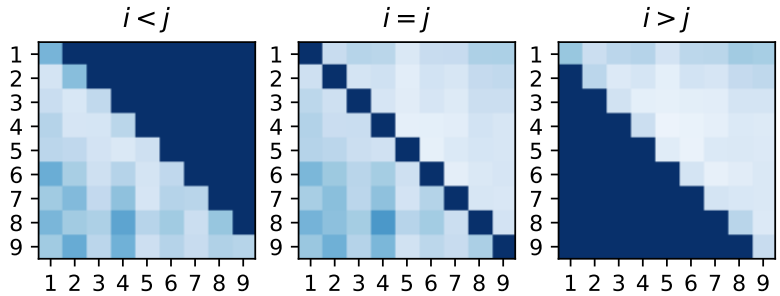

Figure B.1: Probabilities of different (in)equalities according to GPT-3 text-davinci-002, log domain

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.*".

B.3 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"*.

B.4 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

Table C.1: Different alternatives of probabilistic reasoning with **ThinkSum** for solving Odd one out.

Model	List of Words	Latent variable model	Minority voting
ThinkSum davinci-002	0.84	0.67	0.70
ThinkSum davinci-001	0.74	0.77	0.70

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."*

C Additional experiments

ThinkSum vs Auxiliary Knowledge. Table B.1 shows the comparison between auxiliary knowledge and **ThinkSum**. All methods used text-davinci-002. For Phrase relatedness, auxiliary knowledge was generated using the "list differences" prompt shown in Figure 4. For both auxiliary knowledge and **ThinkSum**, 6 number of differences were used, as that was the best for auxiliary knowledge, shown in Figure 3. **ThinkSum** Odd one out and Phrase relatedness are solved with the list of words prompt. For Logical deduction, the **Think** prompt shown in Fig6 was included before the question in the prompt. In all cases, **ThinkSum** outperforms auxiliary knowledge.

Latent variable models for odd one out. While the detection of the odd item can be performed with simple operations, it is also possible to assume that there is a latent structure consisting of two or more clusters such that the facts and items belonging to a cluster can be freely interchanged keeping probability of such combination high. While there are simpler alternative solutions to this task, the latent variable model enables selecting the facts that characterize the majority class, explaining why the minority item is ruled as the odd one out. Thus, expanding on the problem and applying a latent variable model can help interpret the decisions of the system.

More formally, since I includes items that are semantically related, and the odd, we can model $i \in I$ and $f \in F$ to be generated from a latent class $c \in \{0, 1\}$. Then, the conditional distribution can be modeled as:

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

The semantic components, groupings $P(i|c)$ and $p(f|c)$ can be computed from the matrix using expectation-maximization (EM) (Dempster et al., 1977). Then, the score for an item i belonging to a cluster while all other items $m \in S, \{m \neq i\}$ belonging to another cluster can be found as $S_i = \sum_{c,c' \neq c} P(i|c)P(c) \prod_{m \neq i} p(m|c')p(c')$.

We show the effectiveness of the latent variable models in table C.1, 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 run for 200 iterations. All probabilistic reasoning methods perform well, outperforming previous baselines reported in Table 1. Each of these approaches can be applicable in other tasks of similar structure to Odd one out.

Using GPT-3 to evaluate inequalities. Fig. B.1 shows the matrix of posterior probabilities evaluated using GPT-3 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:"* where *"x"* and 'y' are substituted with numbers $\{1, .., 9\}$, and normalizing the probability of the first token over to options *"true"* and *1'false"*. These are the probabilities that statements T_t would evaluate.

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 on Pytorch (Paszke et al., 2019) with a Tesla K80 GPU.

Auxiliary Knowledge	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.
Task	Pick the odd word out: glass, head, arm, leg, hand, foot Answer:

Figure D.1: Auxiliary knowledge prompting applied to ODD ONE OUT. Facts are generated using the "list differences" prompt described in Figure 4, and post-processed according to D.2.

D.1 HYPERPARAMETERS

max_tokens. 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.

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.

D.2 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 concise knowledge statement regarding each object. 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-processsed knowledge statements before the question in the task. An example is illustrated in Figure D.1.

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

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 Answer: They all make noise. Answer: They all are yellow. Answer: They all are binary. Answer: They all go fast. Answer: They all have stripes.
Task Logical Deductions: 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.
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]*