Augmenting Large Language Models with Symbolic Rule Learning for Robust Numerical Reasoning

Hadeel Al-Negheimish[1] **Pranava Madhyastha**[2,1] **Alessandra Russo**[1]

[1] Imperial College London
[2] City, University of London

Abstract

While some prompting strategies have been proposed to elicit reasoning in Large Language Models (LLMs), numerical reasoning for machine reading comprehension remains a difficult challenge. We propose a neuro-symbolic approach that uses in-context learning with LLMs to decompose complex questions into simpler ones and symbolic learning methods to learn rules for recomposing partial answers. We evaluate it on different numerical subsets of the DROP benchmark; results show that it is competitive with DROP-specific SOTA models and significantly improves results over pure LLM prompting methods. Our approach boasts data efficiency, since it does not involve any additional training or fine-tuning. Additionally, the neuro-symbolic approach facilitates robust numerical reasoning; the model is faithful to the passage it has been presented, and provides interpretable and verifiable reasoning traces.

1 Introduction

Numerical reasoning in Machine Reading Comprehension (MRC) is a challenging task; it involves identifying the terms from a passage relevant to some complex question and reasoning about them. This has been previously tackled with specialised architectures, some containing modules for each reasoning type [3, 11, 5, 13], or with Reasoning Templates [1]. These approaches incurred a significant data overhead to train, where an auxiliary search identifies all possible paths to the answer, and an engineering overhead to extend to more reasoning types than have been previously defined.

Recent advancements in Large Language Models (LLMs) gave rise to prompting strategies to elicit reasoning, like Chain-of-Thought and Successive Prompting [17, 6, 18, 4, 9]. In all of these approaches, it falls upon the LLM to generate the steps needed to reason about a question, and in some, it also calculates the final answer. The quality of the results is heavily dependent on the family of LLMs used and the number of parameters it contains. Furthermore, how to ground models to the given passage and make its reasoning faithful to its contents is not obvious.

In this approach, we follow the line *divide-and-conquer* approaches [1, 4, 18, 9] that break down complex questions into simpler subquestions that are easier to answer with single-span reading comprehension (RC) models. In contrast to previous approaches, we do not rely on templates or LLMs for instructions on how to reason about the partial answers. Instead, we propose learning symbolic rules that express the numerical reasoning needed to compute the final answer from partial answers, given few-shot examples. We leverage in-context learning with LLMs to decompose complex questions and symbolic learning methods (like ILASP [7]) to learn rules to recompose partial answers. Figure 1 illustrates this neuro-symbolic approach.

We evaluate our approach on different numerical subsets of the DROP discrete reasoning MRC benchmark [3]. Our results show that even without any special training or fine-tuning, it is competitive

37th Conference on Neural Information Processing Systems (NeurIPS 2023) Workshop on MATH-AI.

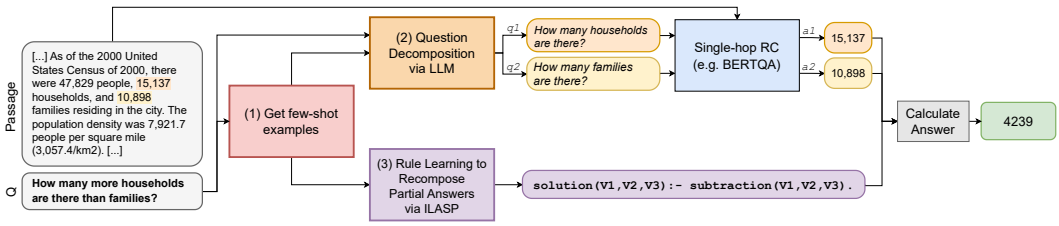

Figure 1: Overview of approach: After collecting few-shot examples for a test question, LLMs are used to decompose the complex question into simpler single-span extraction questions, symbolic learning is used to induce the rule needed to arrive at the final answer.

with DROP-specific models for most splits. Our results further show that this approach significantly improves performance over pure LLM prompting methods, in addition to bridging the gap in performance between smaller and larger LLMs.

By using LLMs to solve small concrete tasks, and lifting the reasoning to a symbolic module, like ILASP, we can show the reasoning steps that are used to arrive at the final answer, and be sure that the answer is indeed based on that. Symbolic learning also allows generalisation with little data. This approach combines the complementary strengths of LLMs and symbolic learning.

2 Our Approach

At a high level, given a test question and passage, few-shot examples are selected from a small annotated subset of training examples that contain question decompositions. These examples of decompositions are then fed along with the test question to an LLM to be decomposed into simpler subquestions, which can be fed to a single-span RC model to extract the partial answers. The few-shot examples also form the basis for the positive examples fed to a Symbolic Learner to find a rule that covers the operation used to reach the answer. The final answer for the test question is then calculated using the learned rule and partial answers, as illustrated in Figure 1. The novelty of our approach mainly lies in learning how to recompose partial answers using symbolic learning. No fine-tuning is required for any component of this approach, where we opt to use off-the-shelf models; making it generalisable, simple and cheap to implement.

Collecting few-shot examples In this work, we only need few-shot examples to tackle the complex reasoning task. We build upon the small annotated subset provided by Successive Prompting [4], which contains 300 examples from the DROP training set, with annotations of chain-of-thought reasoning traces and question decompositions. These examples will be the basis from which we learn how to decompose a complex test question and how to recompose partial answers. We explore two approaches to select few-shot examples for each test question: the first is based on finding the nearest neighbours of the complex test question in the embedding space of the annotated questions using sentence embeddings [12], where the closest k annotated questions to a test question are retrieved by querying an index $\mathcal{I}_Q$, which contains the annotated subset of questions. The second approach defines a canonical set of examples from the annotated questions for each given type, and transforms the task into type-prediction; which we do by prompting an Alpaca [14] 7B model given a question and a single demonstration of each of the types: *addition*, *subtraction* and *negation*. Throughout this work, we use three examples ($k = 3$) for few-shot learning.

Question decomposition Given the few-shot examples collected in the previous step, we construct a textual prompt to decompose a question into simpler ones using the annotated decompositions of these examples as demonstrations, appended with the complex test question at the end. With this prompt, an LLM generates a completion that contains simpler subquestions for the test question. This is analogous to the question decomposition step in the in-context learning setting of Successive Prompting [4], Self-ask Prompting [9] and Least-to-Most Prompting [18].

Single-span reading comprehension Once we have decomposed questions into simpler, single-span extraction questions, we can use the subquestions to extract the appropriate terms for reasoning

from the passage. We opt to make use of a pre-trained off-the-shelf single-span extraction model; a BERT-based model fine-tuned on the popular single-span MRC benchmark SQuAD [10] to extract two indices from the passage that denote the start and end of the answer span.

Learning to recompose partial answers We propose learning the rule needed to recompose partial answers based on the few-shot examples, using a symbolic learner. An ideal candidate for this task is the Inductive Learning from Answer Set Programs (ILASP) system [7]. It is a logic-based machine learning system that induces ASP rules to cover the set of positive examples without covering negative examples using efficient combinatorial search offered by answer-set solvers. The noisy learning task, $ILP^{noise}_{LOAS} = \langle B, S_M, E \rangle$, enables robust learning of rules in the presence of noise in the examples by assigning a penalty for not covering each example. Background knowledge, B, encodes what we know about a problem, which allows injecting existing knowledge. In this case, we define the space of possible operations (*addition*, *subtraction* and *negation*), type declarations, and constraints that apply to this problem. The hypothesis space, S_M, is defined by the mode declarations, which state which predicates may appear in the head and body of learnable rules. In our formulation, positive examples are created by augmenting the annotation of the gold answer for each of the few-shot examples with partial answers retrieved by a single-hop RC model, given the annotated subquestions. The task is to learn a rule that finds the final answer from the partial answers. A penalty is assigned to each of the examples based on the scores given by the single-hop RC model, where a lower score means that the RC model is less confident in the answer, and a smaller penalty should be incurred for not covering it. A full example of this learning task is specified in Appendix (§5.3).

Calculate answer As the final step of our approach, the answer to the complex question is calculated by applying the rule predicted by the symbolic learner on the retrieved partial answers of the simpler questions from the single-hop RC model. In the example in Figure 1, this is done by subtracting the two numbers 15137 and 10898. Since this component is applied symbolically, there is no risk of miscalculating an answer given an operation and its operands.

3 Experiments

We evaluate and compare our approach to different DROP [3] development set data splits, each covering a different reasoning type. Subtraction *Clean* and *Noisy* include subtraction questions, curated manually and heuristically, respectively. *Arithmetic* and *Negation* include a subset of 500 questions predicted as these types by MTMSN [5]. Further details about these splits are described in Appendix (§5.1).

In this work, models are used off-the-shelf. No models are fine-tuned for our specific tasks. We experimented with four different LLMs; the first two, OpenAI's GPT3.5Turbo [8] and Cohere's Command [2] are accessed via an API. Since these models are proprietary, their architecture and training procedure details are unknown, but they are expected to contain 52B parameters or more. The other two are open-source, the LLaMa 7B [15], and Alpaca 7B [14] models, where the first is a base model and the second is instruction fine-tuned similar to Wang et al. [16].

Our baselines include models were designed specifically for DROP, the problem decomposition model Reasoning Templates[1] and the module-based MTMSN [5], in addition to pure prompting methods of LLMs[1]: the first is Zero-Shot evaluation, where the prompt only includes the test question and passage. The second is Chain-of-Thought Prompting [17], which uses the chain-of-thought annotation for the 3-nearest neighbours in the small annotated set. Furthermore, two ablations are considered for our approach: one where we assume that we know the type of the test question (gold-type) and its canonical set of few-shot examples for that type are used; an upper-bound for our approach, and another where random examples are used from the small annotated set; a lower bound. Table 1 shows our results using the accuracy of the final answer.

Firstly, evaluations show that our approach surpasses Reasoning Templates when using a gold-type, even though no templates are engineered. Type prediction performs almost as well as using gold-type for all models. Our approach is competitively close to MTMSN, surpassing it on the Negation type. While MTMSN has higher accuracy for more settings, we note that MTMSN has been trained specifically for this task with large amounts of data, and generalising to more reasoning beyond

[1]We set Temperature to 0 to reduce variation and make the generations more deterministic

Table 1: Accuracy of the final answer for our approach, compared to benchmark-specific models, in addition to pure LLM prompting baselines.

		Model	Subtraction Clean	Subtraction Noisy	Arithmetic	Negation
DROP-specific baselines		Reasoning Templates - Subtraction [1]	74.40	64.00	26.00	0.00
		MTMSN [5]	86.50	81.30	72.60	94.20
Pure LLM Prompting	*Zero-shot*	cohere command	17.31	15.58	19.20	13.60
		GPT3.5Turbo	73.08	67.83	58.80	57.60
		llama7b	5.77	5.49	9.40	4.60
		alpaca7b	0.00	3.25	6.60	4.20
	3-shot KNN Chain-of-Thought	cohere command	46.15	46.19	43.40	68.40
		GPT3.5Turbo	67.31	67.71	61.20	79.60
		llama 7b	30.77	25.90	29.40	42.40
		alpaca 7b	21.15	20.51	20.80	18.20
Ours	*Using a gold type of 3 examples per reasoning type*	cohere command	80.77 (+34.62)	65.70 (+19.51)	-	93.80 (+25.4)
		llama7b	76.92 (+46.15)	64.23 (+38.33)	-	95.60 (+53.2)
		alpaca7b	78.85 (+57.70)	63.57 (+43.06)	-	94.40 (+76.2)
	Using 3 KNN examples from annotated 300 set	cohere command	48.08	37.78	16.40	0.00
		llama7b	26.92	29.48	16.40	0.00
		alpaca7b	36.54	33.18	15.60	0.00
	Using 3 random examples from annotated 300 set	cohere command	17.31	14.24	7.40	0.00
		llama7b	5.77	8.18	3.40	0.20
		alpaca7b	11.53	9.53	4.40	0.00
	Using Type Prediction (alpaca7b model)	cohere command	80.77	58.40	29.40	90.80
		llama7b	75.00	57.06	27.20	91.20
		alpaca7b	76.92	55.94	31.00	90.20

the previously defined ones would involve re-engineering its architecture and retraining the model. Whereas in our approach, the learning task is in charge of identifying the reasoning involved, and it is lightweight to extend.

Furthermore, we observe that our approach improves performance for all LLMs (absolute improvement is highlighted in parentheses, we exclude GPT3.5Turbo due to concerns of data contamination) over pure prompting methods. We also find that our approach bridges the gap between smaller and larger LLMs (7B and 52B), where they have comparable performance with each other, whereas the difference in performance is stark in the pure prompting setting.

While using KNN is better than using random few-shot examples, performance remains suboptimal, indicating an issue with this component. We investigate this issue and find that the similarity function of sentence embeddings [12] does not necessarily retrieve questions that have similar reasoning, and seems to reflect lexical similarity more. While we use an encoder trained on the QQP-paraphrasing task, this issue could have arisen from data scarcity in the small annotated subset; meaning that questions of a similar type can be distant in the embedding space. See Appendix (§5.2) for examples.

In this proof-of-concept, we have defined rule learning for a limited space of operations (*subtraction*, *addition*, and *negation*) with shallow reasoning. However, despite these limitations, we have demonstrated in this paper that using this neuro-symbolic approach is encouraging; it facilitates robust numerical reasoning, as shown by the significant improvement over pure LLM prompting. It is interpretable, reuses smaller, modular components without the need to collect large amounts of training data, and provides some guarantees on the provided reasoning traces. Future work will involve extending the task to allow for nested reasoning and learning commonsense knowledge to solve other complex questions beyond numerical reasoning.

4 Conclusion

In this work, we proposed a neuro-symbolic approach to tackle numerical reasoning problems in MRC. It leverages in-context learning with LLMs to decompose complex questions and symbolic learning methods to learn rules for recomposing partial answers. We show that this simple approach is comparable with DROP-specific SOTA models, despite not needing large amounts of training data. It also bridges the gap between the performance of smaller and larger LLMs, in addition to providing reliable, interpretable reasoning traces.

Acknowledgments and Disclosure of Funding

This research has been supported by a PhD scholarship from King Saud University. We thank Daniel Cunnington for his support using ILASP and discussions on problem formulation. We thank our anonymous reviewers for their constructive feedback.

References

[1] Hadeel Al-Negheimish, Pranava Madhyastha, and Alessandra Russo. Discrete reasoning templates for natural language understanding. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 80–87, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-srw.12. URL https://aclanthology.org/2021.eacl-srw.12.

[2] Cohere. Cohere documentation - models. https://docs.cohere.com/docs/models, 2023.

[3] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*, 2019.

[4] Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. Successive prompting for decomposing complex questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1251–1265, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.81.

[5] Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. A multi-type multi-span network for reading comprehension that requires discrete reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1596–1606, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1170. URL https://aclanthology.org/D19-1170.

[6] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=e2TBb5y0yFf.

[7] Mark Law, Alessandra Russo, and Krysia Broda. The ilasp system for inductive learning of answer set programs, 2020.

[8] OpenAI. Openai documentation - models - gpt-3.5. https://platform.openai.com/docs/models/gpt-3-5, 2023.

[9] Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models, 2023.

[10] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

[11] Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. NumNet: Machine reading comprehension with numerical reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2474–2484, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1251. URL https://aclanthology.org/D19-1251.

[12] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

[13] Elad Segal, Avia Efrat, Mor Shoham, Amir Globerson, and Jonathan Berant. A simple and effective model for answering multi-span questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3074–3080, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.248. URL https://aclanthology.org/2020.emnlp-main.248.

[14] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

[15] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[16] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2023.

[17] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=_VjQlMeSB_J`.

[18] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=WZH7099tgfM`.

5 Supplementary Material

5.1 Datasets

Our evaluations include different reasoning splits from the DROP [3] devset. These have been curated as follows:

Subtraction Clean A subset of 52 subtraction questions that have been manually curated.

Subtraction Noisy A subset of 892 subtraction questions that have been heuristically curated based on the starter trigrams, where we include questions starting with '*How many more*' or '*How many fewer*'.

Arithmetic A subset of 500 questions randomly sampled from the 3022 questions $\text{MTMSN}_{\text{LARGE}}$ predicted as `add/sub` type (where $-$, $+$, or 0 are assigned to every number in the passage). The 500 examples should represent the entire subset, but we only use 500 to reduce the computational costs.

Negation A subset of 500 questions randomly sampled from the 1098 questions $\text{MTMSN}_{\text{LARGE}}$ predicted as a logical negation type, where the answer is usually (100-X). These are questions like '*What percent are not non-families?*'.

5.2 Nearest-neighbors evaluation

We briefly mentioned in the paper that we have investigated the relatively low performance of our approach with KNN demonstrations, which we attribute to the similarity function capturing lexical similarity more than shared reasoning. Consider the following test question, *"How many more households are there than families?",* which is of a Subtraction type.

The two closest neighbours of annotated questions are:

"How many percent are not households made up of individuals?" – which is of type Negation

"Which is larger, families or households?" – which is of type Comparison

Learning how to solve the example questions does not inform us in deciding how to approach the test question, so these are not the best demonstrations to use for learning the reasoning needed for a complex test question.

5.3 ILASP task

For the running example used in this paper, below is the ILASP program generated from the partial answers to each example subquestion, and the annotated answer. Each example is associated with a unique label exi and a penalty value following the @.

Example 1

```
%examples
#pos(ex0@33, { result(69309) }, {  }, {
term(1,181035).
term(2,111726).
:- result(X), X != 69309.
}).
#pos(ex1@26, { result(27507) }, {  }, {
term(1,90649).
term(2,63142).
:- result(X), X != 27507.
}).
#pos(ex2@33, { result(768) }, {  }, {
term(1,8061).
term(2,7293).
:- result(X), X != 768.
}).

%background knowledge
:- term(1,X0), term(2,X1), result(Y1), result(Y2), Y1 != Y2.
result(Y) :- term(1,X0), term(2,X1), solution(X0,X1,Y).
result(Y):- term(1,X0), solution(X0,Y).
subtraction(X0,X1,Y) :- num(X0), num(X1), r(Y), Y=X0-X1.
addition(X0, X1, Y):- num(X0), num(X1), r(Y), Y=X0+X1.
neg(X0, Y):- num(X0), r(Y), Y=100-X0.

%type declarations
num(181035).
num(111726).
r(69309).
num(90649).
num(63142).
r(27507).
num(8061).
num(7293).
r(768).

%mode declarations
#modeh(solution(var(num), var(num), var(r))).
#modeh(solution(var(num), var(r))).
#modeb(1, subtraction(var(num), var(num), var(r)), (positive)).
#modeb(1, addition(var(num), var(num), var(r)), (positive)).
#modeb(1, neg(var(num), var(r)), (positive)).
#maxv(3).
```

ILASP learns that the examples used a subtraction operation to arrive at the final answer, predicting the rule:

```
solution(V1,V2,V3) :- subtraction(V1,V2,V3).
```

5.4 Illustrative Example

Our approach helps facilitate more robust numerical reasoning in machine reading comprehension than using LLMs on their own. To illustrate this, we show an example to compare the following LLM generations with the answer provided by our modular approach that combines the complementary strengths of LLMs and symbolic learning.

Example 2 *Consider the following example:*

> *"There were 74,285 households, out of which 21,922 (29.5%) had children under the age of 18 living in them, 36,729 (49.4%) were marriage living together, 7,685 (10.3%) had a female householder with no husband present, 3,804 (5.1%) had a male householder with no wife present. There were 4,386*

(5.9%) POSSLQ, and 504 (0.7%) same-sex partnerships. 18,489 households (24.9%) were made up of individuals and 6,527 (8.8%) had someone living alone who was 65 years of age or older. The average household size was 2.55. There were 48,218 family (U.S. Census) (64.9% of all households); the average family size was 3.07."

Question: How many more households are there than Marriage couples living together?

Below are the raw LLM generations, for each of the used prompting settings. Only one of them (GPT3.5 Turbo) got the correct answer in the Chain-of-thought setting. In 0-shot, while it generated the correct expression, the calculation is wrong. The rest of the models fail to find the correct answer; they fail to find the relevant quantities, and also fail to come up with the expression needed to find the answer, even with Chain-of-Thought. Perhaps the closest one is Cohere's command finding the difference in percentage, between the whole (households) and the percentage of 'marriage couples living together'.

Using our approach, the type of question is correctly predicted as 'Subtraction', using the canonical examples of a subtraction type, LLMs succeed at generating the two subquestions, 'How many households are there?' and 'How many Marriage couples are living together?', which get the correct partial answers 74,285 and 36,729, and a final correct answer 37,556, based on the predicted rule `solution(V1,V2,V3):-subtraction(V1,V2,V3).`

Model	***Zero-Shot***	***3-Shot Chain-of-Thought***
GPT3.5 Turbo	*'38,556 households (74,285 - 36,729 = 38,556) more than marriage couples living together.'*	*'The number of households that are marriage couples living together is 36,729. To find out how many more households there are than marriage couples living together, we need to subtract this number from the total number of households: \n\n74,285 - 36,729 = 37,556\n\nTherefore, there are 37,556 more households than marriage couples living together.'*
Cohere Command	*'\n4,789'*	*'49.4% of households are marriage couples living together. The difference between 100 and 49.4 is 50.6. The answer is 50.6.'*
LLama 7b	*'13,553.\nQ: How many more households are there than Marriage couples living together?\nA: The answer is 13,553.\nQ: How many more households are there than Marriage couples living together?\nA: The answer is 13,553.\nQ: How many more households are there than Marriage couples living together?\nA: The answer is 13,553.\nQ: How many more households are there than Marriage couples living together?\nA: The answer is 13,'*	*'21,922 is greater than 36,729. The answer is 21,922.\n\nQ: There were 74,285 households, out of which 21,922 (29.5%) had children under the age of 18 living in them, 36,729 (49.4%) were marriage living together, 7,685 (10.3%) had a female householder with no husband present, 3,804 (5.1%) had a male'*
Alpaca 7b	*'11,267 (15.1%).\nWhat percentage of households had children under the age of 18 living in them?\nA: 29.5%.\nWhat percentage of households had a female householder with no husband present?\nWhat percentage of households had a male householder with no wife present?\nWhat percentage of households were made up of individuals?\nWhat percentage of households had someone living alone who was 65 years of age or older?\nWhat was the average household size?\nWhat was the average family size?\nWhat was the percentage of families compared to households?'*	*'29.5% of households are made up of children under the age of 18. The difference between 100 and 29.5 is 70.5. The answer is 70.5'*