Equipping Language Models with Tool Use Capability for Tabular Data Analysis in Finance

Adrian Theuma **Ehsan Shareghi**
Department of Data Science & AI, Monash University
adriantheuma@gmail.com ehsan.shareghi@monash.edu

Abstract

Large language models (LLMs) have exhibited an array of reasoning capabilities but face challenges like error propagation and hallucination, particularly in specialised areas like finance, where data is heterogeneous, and precision is paramount. We explore the potential of language model augmentation with external tools to mitigate these limitations and offload certain reasoning steps to external tools that are more suited for the task, instead of solely depending on the LLM's inherent abilities. More concretely, using financial domain question-answering datasets, we apply supervised fine-tuning on a LLAMA-2 13B CHAT model to act both as a *task router* and *task solver*. The *task router* dynamically directs a question to either be answered internally by the LLM or externally via the right tool from the tool set. Our tool-equipped SFT model, RAVEN, demonstrates an improvement of 35.2% and 5.06% over the base model and SFT-only baselines, respectively, and is highly competitive with strong GPT-3.5 results. To the best of our knowledge, our work is the first that investigates tool augmentation of language models for the finance domain.[1]

1 Introduction

Augmenting Large Language Models (LLMs) with tools has emerged as a promising approach to further complement LLMs' capabilities with specialised mechanisms, leading to improved accuracy and reliability (Schick et al., 2023; Yao et al., 2023). This approach offloads tasks, fully or partially, to a deterministic offline tool such as a python interpreter (Gao et al., 2023), calculator (Cobbe et al., 2021), knowledge base (Borgeaud et al., 2022), or online APIs of models and services (Yao et al., 2023; Qin et al., 2023; Shen et al., 2023).

This paradigm holds particular appeal in fields demanding precision, such as finance (Yang et al., 2023) and healthcare (Luo et al., 2022; Singhal et al., 2022). Specifically, the specialised terminology within the finance domain and the diverse range of data sources, encompassing both structured and unstructured data, along with the complex numerical reasoning requirements across such heterogeneous sources, render it an ideal candidate for potential improvements through tool augmentation. Nevertheless, there has been limited research dedicated to this specialised domain.

A satisfying review of existing works on tool augmentation of LLMs is beyond the scope of this work; however, this space can be divided into two primary directions: (1) approaches that require an LLM at the center and uses few-shot in-context learning to either provide tool and API documentations, or demonstrations that involve tool use (Hsieh et al., 2023; Qin et al., 2023; Shen et al., 2023; Hsieh et al., 2023), and (2) approaches that build fine-tuned smaller LMs under a static tool use protocol (Schick et al., 2023), or through expensive annotations collected from commercial LLMs (Chen et al., 2023; Yao et al., 2023).

In this work, our primary focus lies in demonstrating the potential of tool augmentation within the finance domain. Acknowledging the utmost significance of privacy concerns within the financial sector, we have chosen to adopt a fully offline approach, equipping a language model with diverse tool utilisation mechanisms. More concretely, we employ Parameter Efficient Fine-Tuning (PEFT) (Hu et al., 2022; Houlsby et al., 2019) to equip a LLAMA 2 13B CHAT (Touvron et al., 2023) with tool use capabilities. Our approach differs from previous research in two significant ways. First, we do not rely on costly annotations of training examples produced by commercial language models. Second, we enhance existing question-answering training datasets by incorporating instructions and merge data representing various tasks. This approach instructs the model to adapt dynamically and determine the most appropriate mechanism (either internal or tool-based) to address each specific query.

[1] Code, model, and data: https://raven-lm.github.io

Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics
Volume 2: Short Papers, pages 90–103

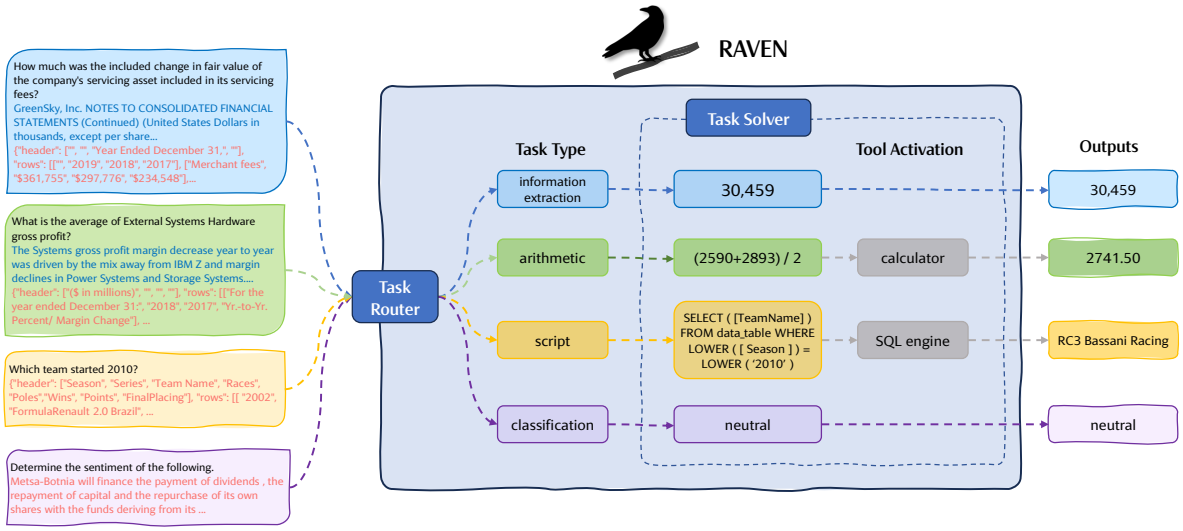

Figure 1: **RAVEN Inference Flow**. Using the language model the *Task Router* infers the optimal task format to use conditioned on the given prompt. The *Task Solver* re-formats the instruction according to the selected template by the task router and sends it to the language model again. The pipeline will branch between serving the response directly or calling a tool API to perform an intermediate evaluation before serving the final output.

Our model, RAVEN, achieves significant improvements in reasoning over structured data. For example, compared to the base model we demonstrate a lift in exact match accuracy of 63.8% (21.68% → 85.52%) on the WIKI-SQL (Xu et al., 2018). Despite being much smaller in size, RAVEN also outperforms GPT-3.5 on all benchmarks with an absolute average accuracy lift of 9.2%.

2 RAVEN

We use the LLAMA 2 13B CHAT (Touvron et al., 2023) model as the backbone and fine-tune it using Low Rank Adaptation (LoRA) (Hu et al., 2022). In this section we provide training details of RAVEN. The overall architecture of RAVEN is shown in Figure 1.

2.1 Fine-tuning Data

We use a mixture of four financial and generic structured and unstructured question-answering datasets. We provide a brief summary in below.

TAT-QA. Consists of questions generated by financial experts associated with hybrid contexts drawn from real-world financial reports (Zhu et al., 2021). The questions typically require a range of data extraction and numerical reasoning skills, including multiplication, comparison, sorting, and their various combinations. Apart from the answer, TAT-QA also provides the derivation, where applicable, which proves beneficial for offloading the calculation to an external tool, as will be explained in §2.2.

Financial PhraseBank. Consists of phrases derived from English news on listed companies in OMX Helsinki (Malo et al., 2014). The dataset contains phrase-level annotation by financial markets experts, that categorise each sample sentence as either positive, negative, or neutral, from an investor's standpoint. This dataset is relevant because sentiment analysis models trained on general datasets do not perform well in specialised domains due to the unique vocabulary found in financial texts, which often do not rely on easily identifiable positive or negative words (Araci, 2019).

Wiki-SQL. Consists of manually annotated crowd sourced examples of natural language questions and SQL queries over tables found on Wikipedia (Zhong et al., 2017). Whilst this is not specifically a financial domain dataset its relevancy is in the availability of the script that produces the answer. Similar to the derivation in the TAT-QA dataset this script is crucial to steer our model to use a tool instead of producing the answer directly.

OTT-QA. Similar to TAT-QA, this dataset consists of questions over tabular data and unstructured text across diverse domains (Chen et al., 2021). The majority of questions necessitate multi-hop inference involving both forms of data. The dataset's relevance lies in its omission of derivation or intermediate steps, which poses a challenge for the model to infer the correct answer.

Data splits. Among the four datasets, FPB[2] and

[2] https://github.com/vrunm/Text-Classification-Financial-Phrase-Bank

OTT-QA[3] lack a published test split. TAT-QA[4] has a test split without gold labels. WikiSQL[5] provides a public test set. We used the WikiSQL test split, and for the other 3 datasets generated random 80-10-10 splits (available here). Table 1 summarises the statistics of the datasets.

2.2 Tools

RAVEN is equipped with two external offline tools: a calculator and a SQL engine. The *Calculator* is instantiated in a python interpreter and is used to evaluate well-formed arithmetic expressions. The API expects one input representing the arithmetic expression and returns the evaluated result. The *Lightweight SQL engine* is an API capable of executing SQL scripts on relational data. The API expects two inputs, (1) a string representation of the structured data and (2) a SQL script. The API's lightweight database engine converts structured data from its textual form to the engine's relational representation and converts data types where applicable. The SQL script is executed on this representation and the API returns the result.

2.3 Instruction Tuning

Inspired by Wang et al. (2023) and Taori et al. (2023) we engineer various templates for SFT instruction tuning. In general, we require to extract up to four key attributes from the original datasets. These are (1) *instruction* that describes the task to perform, for example, "*Determine the sentiment of the following phrase*", or the question "*What is the percentage change in revenue after the adoption of ASC 606?*" (2) *input* that provides more context such as the phrase to classify or a passage, (3) *data* that accompanies the context in tabular format, (4) *derivation* that produces the answer or expected *response*. The instruction and one of derivation or response are mandatory, whilst the other attributes are included if applicable.

To ensure training diversity, our model is trained on a combination of all available training data. Based on the data, we craft different templates depending on which tool the model should choose or if the model should directly answer the question on its own (i.e., to train the *Task Solver* in Figure 1). We also automatically generate another dataset, that supplements the above question-answer dataset for training our model to select the appropriate template based on the context (i.e., to train the *Task Router* in Figure 1). Refer to appendix C for template examples.

2.4 Inference

During inference, we follow a two-step process with RAVEN. First, we employ a specialised *template choice* prompt to determine the most suitable prompt template (from "arithmetic," "classification," "script," or "information extraction") based on the input. Next, we wrap the instruction, including the input and relevant data, in the inferred prompt template and send it to RAVEN for generating the subsequent output. Depending on the selected template, the *Task Solver* either activates a tool to fulfil the request or directly produces the response.

We discuss the inference behaviour when each of these templates are used. For **Script** the model is expected to produce a well-structured SQL script. In this scenario, the structured data table provided in the prompt is temporarily loaded in memory using a lightweight database engine, and the script execution on the table produces the output. For **Arithmetic** the model is expected to predict a well formed arithmetic expression. This expression is evaluated by a calculator and the resulting value passed as output. The **Information Extraction** template instructs the model that there is information included in structured form that needs to be considered before producing the answer. In this case no tool is used and the model is expected to infer the correct output based solely on the information in the prompt. The **Classification** template is used when the prediction of the model should be taken as-is.

3 Experiments

We compare with the base LLAMA 2 13B CHAT with and without SFT[6]. We also report GPT-3.5[7] (5-shot), GPT-3.5 (Chain-of-Thought (Wei et al., 2022)) and GPT-3.5 (5-shot + Tools). The SFT model trained with tool use is denoted as RAVEN. When tool use fails due to ill-formed arguments we have a fallback mechanism to produce the answer by the SFT model, denoted as BACKOFF. For training details and hardware, see Appendix B. We evaluate the models using *exact match*. The task router has determined the correct type 100% of the time, except for TAT-QA where the accuracy was 90.62%.

[3] https://github.com/wenhuchen/OTT-QA
[4] https://nextplusplus.github.io/TAT-QA/
[5] https://github.com/salesforce/WikiSQL

[6] To steer the base model into producing a short answer we add *"Output the answer only with no explanation."* to the prompt.
[7] gpt-3.5-turbo

	Statistics			Models						
Dataset	Train	Dev	Test	GPT-3.5 (CoT)	GPT-3.5 (5-shot)	+Tools	Llama2	+SFT	Raven	+Backoff
TAT-QA	10,477	1,162	1,278	19.23%	34.06%	46.82%	10.91%	37.87%	51.35%	**52.27%**
OTT-QA	10,273	1,115	1,247	5.55%	14.55%	14.60%	6.18%	**20.10%**	16.03%	16.03%
Wiki-SQL	12,782	1,391	1,536	32.07%	53.00%	75.88%	21.68%	74.38%	84.25%	**85.52%**
FPB	3,413	382	421	44.18%	70.07%	71.73%	66.03%	90.97%	**91.92%**	91.92%

Table 1: The data statistics and experimental results (Exact Match) of various benchmarks and models. The best results are in **bold**. GPT-3.5 results are based on 5-shots. SOTA is based on previously published results.

3.1 Main Results

The results are summarised in Table 1. Compared to the base model, RAVEN significantly improves the results on the **PhraseBank** dataset by an absolute 25.9%. On the **Wiki-SQL** dataset the base model is *unable* to infer the correct answer almost 80% of the time. This figure is inverted for RAVEN which obtains a *4-fold* improvement over the base model inferring the correct answer more than 85% of the time. Our model improves on the best GPT-3.5 performance by close to 10% (absolute). All the questions in this dataset can be addressed using the lightweight database engine and involve a combination of data selection, ranking and arithmetic operations on structured data. This result underscores the distinct advantage of delegating this task to a tool rather than relying on the language model to infer the results in a zero-shot manner. Despite the results not being as strong as RAVEN we observe a similar pattern on the GPT-3.5 evaluation in which better results are incrementally obtained when including examples in the context and using tools compared to CoT.

We see a similar pattern on the **TAT-QA** benchmark with the tool augmented model achieving a *5-fold* improvement on the base model. Approximately 46% of the observations of the TAT-QA dataset are annotated with an intermediate arithmetic derivation that RAVEN evaluates using a calculator at inference time. We perform a comparative analysis to explore whether our model performs better on this portion of the data in the analysis section (§3.2).

In **OTT-QA**, the majority of questions require multi-hop inference involving both tabular data and unstructured text, with the information needed to answer the questions dispersed across these two input types. This dataset does not have annotated intermediate steps to get to the answer and therefore all models are expected to infer the answer without relying on tools. Despite SFT achieving an increase in accuracy compared to the base model, the relatively low score underscores the importance of intermediate reasoning steps and tools (Chen et al., 2023).

We observed the BACKOFF mechanism to bring slight improvement on TAT-QA (51.35% → 52.27%) and WIKI-SQL (84.25% → 85.52%).

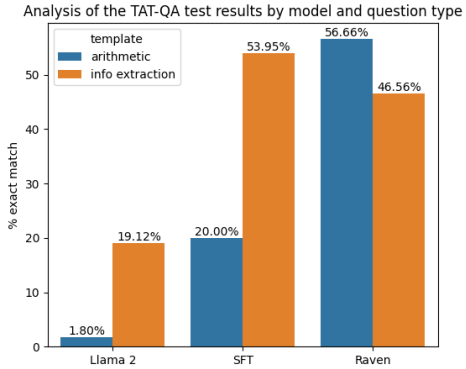

Figure 2: Comparison of model performance on the TAT-QA dataset specifically highlighting the effect of a tools-augmented model on questions that require multi-hop reasoning.

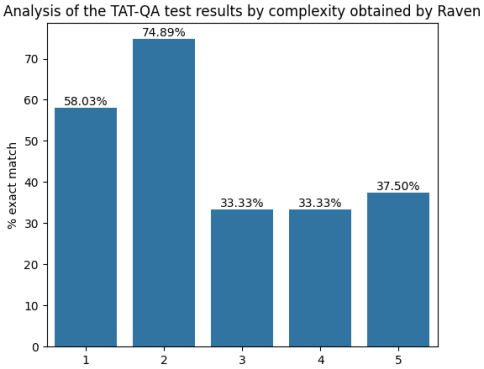

Figure 3: Comparison of model performance on the TAT-QA dataset highlighting the effect of complexity on model performance.

3.2 Analysis

Is it better to have a separate model for each task? We developed a model specifically using the TAT-QA dataset, achieving an evaluation score of 54.70%. This dedicated model outperforms RAVEN by 2.4%. We contend that this modest per-

formance gain does not warrant the added complexity of maintaining separate models and switching between them during inference.

Why tool augmentation is necessary? Approximately half of the questions within the TAT-QA dataset are annotated with an arithmetic equation. The presence of the equation implies that the language model needs to perform multiple actions to output the correct answer. This process involves the correct extraction of, at a minimum, two numerical values from the context, followed by the execution of an arithmetic operation, such as addition or division. This particular scenario is ideal to understand the effect of SFT and tool augmentation by comparing the performance of different models on the two categories of data from the same dataset. As shown in Figure 2 the base model without any fine-tuning is ill-equipped to perform multi-hop reasoning achieving close to 2% accuracy equating to ten correct answers of approximately 620. Although we observe an improvement in the SFT model, the impact of using tools is evident in the substantial jump to 56.7% accuracy achieved by RAVEN. These findings further confirm that SFT models are able to accurately extract multiple data points from the context but require external tools to correctly compose the final answer from the gathered data. This is also evidenced by the consistent performance of the *Information Extraction* type questions between SFT and RAVEN, which only requires data extraction to answer the question.

The utility of augmenting language models with external tools is substantiated further through a comparative analysis of experimental outcomes on two similar datasets. Addressing questions on WIKI-SQL and OTT-QA requires multi-hop reasoning across diverse forms of data, spanning both structured and unstructured formats. The primary difference lies in the annotation method: the WIKI-SQL dataset is annotated with a data extraction script which, when executed on the structured data, yields the answer. In contrast, the OTT-QA dataset lacks this intermediate derivation step. By delegating the script execution to an external tool, RAVEN achieves an exact match accuracy of 85.52% on WIKI-SQL and 16.03% on OTT-QA, underscoring the effectiveness of fit-for-purpose external tools in this scenario.

What is the impact of question complexity? On the TAT-QA dataset we can use the number of arithmetic operators in the *gold* arithmetic equation as a proxy for question complexity. One arithmetic operator implies the extraction of two numerical values from the context, two operators, three numerical values, and so on. As shown in Figure 2, RAVEN's performance degrades with the number of numerical values to extract from the context.

4 Conclusion

In this paper we have demonstrated the feasibility of equipping a LLAMA 2 13B CHAT model with tool use capabilities via fine-tuning a mere 0.2% of its parameters on a relatively small and diverse dataset. The augmentation with tools remarkably elevated the performance of the base model by an average of 35.2% across 4 datasets, surpassing even a significantly larger GPT-3.5 model by 9.2%. Additionally, through a comparative analysis of question answering datasets we demonstrate the effectiveness of augmenting language models with external tools, showing significant improvements in accuracy when addressing multi-hop questions with tools.

Limitations

Infrastructure Bottleneck. Our experiments were constrained with fitting our model on available commodity hardware. We hypothesise that it would be possible to obtain better performance using the larger LLAMA 2 70 billion-parameter model and a longer context length. Experiments by Touvron et al. (2023) demonstrated that the 70-billion-parameter model consistently achieves the highest performance across various prominent natural language understanding benchmarks. Additionally, a longer context length enables experimentation with diverse prompts as well as alternative representations of structured data.

Language model evaluation. Free-form natural language generation (NLG) poses significant evaluation challenges that remain under-studied to this date (Liu et al., 2023). Zheng et al. (2023) argue that while users prefer the responses of an instruction-tuned model over the base model, traditional LLM benchmarks (Liang et al., 2022; Hendrycks et al., 2021) cannot tell the difference. This challenge is heightened in specialised domains such as finance. Common similarity scores such as BLEU (Papineni et al., 2002) which measures *n-gram* overlap between candidate and reference sentences are unsuitable due to misleading accuracy or penalised semantic correctness (Freitag et al., 2022). Although BERTSCORE (Zhang et al., 2020) addresses some of these pitfalls by measuring the similarity of candidate and reference sentences using pre-trained contextualised embed-

dings it can still produce high scores for inaccurate results. For example the candidate and reference sentences *"The amount of goodwill reallocated to the IOTG operating segment in 2018 was $480* ***million****"*, and *"The amount of goodwill reallocated to the IOTG operating segment in 2018 was $480"* have a BERTSCORE (f1) of 99.17%! These measures are not suitable for comparing numerical content.

Conversely, using exact match criteria might unjustly penalise NLG models, given that identical numerical values can be expressed in varying forms - such as "$4 million" and "$4,000,000," or "0.24" and "24%,". In some cases, numerical values can be integrated within a passage of text, rendering the evaluation of such content very challenging. In our evaluation we have normalised different formatting (such as converting values to percentages where appropriate), however a universal normalising algorithm in this space is outside the scope of our research.

GPT-3.5 evaluation. Evaluating our benchmark with GPT-3.5 poses significant challenges, especially when using ZERO-SHOT (COT) (Kojima et al., 2022). GPT-3.5 does not consistently adhere to instructions for providing a concise response, such as a single word or number, which makes *exact match* comparisons challenging. Additionally, we have noticed that GPT-3.5 does not generate a response when uncertain. This is particularly evident when evaluating the FPB, which does not exhibit common sentiment negative or positive words.

Ethics Statement

Our work is built on top of existing pre-trained language models. Our goal was not to attend to alleviate the well-documented is- sues (e.g., privacy, undesired biases, etc) that such models embody. For this reason, we share the similar potential risks and concerns posed by these models. Additionally, our SFT was conducted on publicly available research benchmarks, and as such the additional SFT step used in RAVEN is unlikely to introduce any new area of risk.

References

Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *CoRR*, abs/1908.10063.

Jillian Bommarito, Michael J. Bommarito II, Daniel Martin Katz, and Jessica Katz. 2023. GPT as knowledge worker: A zero-shot evaluation of (AI)CPA capabilities. *CoRR*, abs/2301.04408.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. 2023. Fireact: Toward language agent fine-tuning. *arXiv preprint arXiv:2310.05915*.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W. Cohen. 2021. Open question answering over tables and text. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

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

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale. *CoRR*, abs/2208.07339.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Markus Freitag, Ricardo Rei, Nitika Mathur, Chi-kiu Lo, Craig Stewart, Eleftherios Avramidis, Tom Kocmi, George F. Foster, Alon Lavie, and André F. T. Martins. 2022. Results of WMT22 metrics shared task: Stop using BLEU - neural metrics are better and more robust. In *Proceedings of the Seventh Conference on Machine Translation, WMT 2022, Abu Dhabi, United Arab Emirates (Hybrid), December 7-8, 2022*, pages 46–68. Association for Computational Linguistics.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. PAL: program-aided language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764–10799. PMLR.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Cheng-Yu Hsieh, Si-An Chen, Chun-Liang Li, Yasuhisa Fujii, Alexander Ratner, Chen-Yu Lee, Ranjay Krishna, and Tomas Pfister. 2023. Tool documentation enables zero-shot tool-usage with large language models. *CoRR*, abs/2308.00675.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

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

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel J. Orr, Lucia Zheng, Mert Yüksekgönül, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2022. Holistic evaluation of language models. *CoRR*, abs/2211.09110.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using GPT-4 with better human alignment. *CoRR*, abs/2303.16634.

Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. 2022. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings Bioinform.*, 23(6).

Pekka Malo, Ankur Sinha, Pekka J. Korhonen, Jyrki Wallenius, and Pyry Takala. 2014. Good debt or bad debt: Detecting semantic orientations in economic texts. *J. Assoc. Inf. Sci. Technol.*, 65(4):782–796.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *CoRR*, abs/2302.04761.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Kumar Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Schärli, Aakanksha Chowdhery, Philip Andrew Mansfield, Blaise Agüera y Arcas, Dale R. Webster, Gregory S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle K. Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. 2022. Large language models encode clinical knowledge. *CoRR*, abs/2212.13138.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti

Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David S. Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *CoRR*, abs/2303.17564.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. Sql-to-text generation with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 931–936. Association for Computational Linguistics.

Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023. Fingpt: Open-source financial large language models. *CoRR*, abs/2306.06031.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *CoRR*, abs/2306.05685.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3277–3287. Association for Computational Linguistics.

A Background on LMs in Finance

Araci (2019) tackles financial sentiment analysis by further pre-training BERT (Devlin et al., 2019) on a financial corpus and uses the resulting sentence embeddings to obtain higher text semantic representation before training a downstream sentiment classifier. The author concludes that fine-tuning the generic language model captures the nuances of financial terminology demonstrated by the absolute SOTA improvement of 15%. Bommarito et al. (2023) use the TEXT-DAVINCI-003[8] API to assess whether LLMs have the potential to augment industry knowledge workers. In agreement with earlier findings (Nye et al., 2021), Bommarito et al. (2023)'s model under-performs human performance by a large margin on quantitative reasoning tasks of the American Institute of Certified Public Accountants (AICPA) assessment while approaching human levels on multiple choice questions, achieving an accuracy rate of 14.4% and 57.6% respectively. Wu et al. (2023) train a 50 billion parameter LLM using Bloomberg's closed source datasets and general-purpose data to obtain BloombergGPT, the first large scale specialised language model in the finance domain. The resulting model performs well on financial benchmarks while retaining general-purpose performance comparable to other foundational models.

B Training Details

Training details. We use the pre-trained weights of LLAMA 2 13B CHAT (Touvron et al., 2023) for the base model and LLAMATOKENIZER for prompt tokenisation. We limit the maximum context length to 1,204 tokens and discard any training observations that exceed this limit after tokenisation. Due to hardware constraints we use a per device train batch of one and accumulate the gradient for 128 steps achieving the equivalent `batch_size` of 128 and use quantisation to load the model in `8-bit` (Dettmers et al., 2022). We adapt the same `optimiser`, `learning_rate` and `warmup_steps` as Taori et al. (2023), and set these to `adamw`, 3×10^{-4} and 100, respectively. We use Low Rank Adaptation to reduce the number of trainable parameters and similar to Taori et al. (2023) set the `rank` and `alpha` hyper-parameters to 16, `dropout` to 0.05 and target the `q_proj`, `k_proj`, `v_proj`, and `o_proj` modules of the base model. This reduces the trainable parameters to 26,214,400 or 0.2% of the base model. The final models are trained for 5 `epochs` totalling 1,200 `steps`.

Training hardware. We train the models on commodity hardware equipped with a 13th Gen Intel(R) Core(TM) i7-13700KF CPU at 3.40 GHz, 64 GB installed RAM and NVIDIA GeForce RTX 4090 GPU with 24 GB onboard RAM. The final model consumed 100 GPU hours during training and 10 GPU hours for evaluation.

Carbon footprint. Given we train two models and an average consumption of 400 Wh we estimate the total power consumption to be 88 kWh with a carbon dioxide equivalent (CO_{2e}) emissions of 0.081 tonnes[9]. To obtain a realistic measure of emissions we also need to consider multiple training experiments with different settings leading to the final models including with different hyper-parameters, prompt templates and other mix of datasets. We estimate the realistic total consumption and emissions is 10-fold that of the final models.

GPT-3.5 Experiments We compare our results with GPT-3.5 using few-shot in-context learning. We use the following *system* to steer the model into producing a short answer. "*You are a data expert that can reason over structured and unstructured data. Use the following examples to help you reason over the final question. Follow the same format of the examples to answer the final question. Output a short response with the answer only and do not include any explanations or introductory sentences.*"

C Templates

Below are a few examples of prompts generated from the datasets used to train RAVEN.

C.1 TAT-QA

Example 1 - The response is an equation
Below is an instruction that describes a task, coupled with input and data providing additional context.

[8] https://platform.openai.com/docs/models/gpt-3-5
[9] https://carbonpositiveaustralia.org.au/carbon-footprint-calculator

Formulate an arithmetic equation to generate the answer.

Instruction:
What was the change in the basic net earnings per share between 2017 and 2019?

Input:
(5) Earnings Per Share Basic earnings per share is computed by dividing Net earnings attributable to Black Knight by the weighted-average number of shares of common stock outstanding during the period. For the periods presented, potentially dilutive securities include unvested restricted stock awards and the shares of BKFS Class B common stock prior to the Distribution. For the year ended December 31, 2017, the numerator in the diluted net earnings per share calculation is adjusted to reflect our income tax expense at an expected effective tax rate assuming the conversion of the shares of BKFS Class B common stock into shares of BKFS Class A common stock on a one-for-one basis prior to the Distribution. The effective tax rate for the year ended December 31, 2017 was (16.7)%, including the effect of the benefit related to the revaluation of our net deferred income tax liability and certain other discrete items recorded during 2017. For the year ended December 31, 2017, the denominator includes approximately 63.1 million shares of BKFS Class B common stock outstanding prior to the Distribution. The denominator also includes the dilutive effect of approximately 0.9 million, 0.6 million and 0.6 million shares of unvested restricted shares of common stock for the years ended December 31, 2019, 2018 and 2017, respectively. The shares of BKFS Class B common stock did not share in the earnings or losses of Black Knight and were, therefore, not participating securities. Accordingly, basic and diluted net earnings per share of BKFS Class B common stock have not been presented. The computation of basic and diluted earnings per share is as follows (in millions, except per share amounts):

Data:
{"header": ["", "", "Year ended December 31,", ""], "rows": [["", "2019", "2018", "2017"], ["Basic:", "", "", ""], ["Net earnings attributable to Black Knight", "$108.8", "$168.5", "$182.3"], ["Shares used for basic net earnings per share:", "", "", ""], ["Weighted average shares of common stock outstanding", "147.7", "147.6", "88.7"], ["Basic net earnings per share", "$0.74", "$1.14", "$2.06"], ["Diluted:", "", "", ""], ["Earnings before income taxes and equity in losses of unconsolidated affiliates", "", "", "$192.4"], ["Income tax benefit excluding the effect of noncontrolling interests", "", "", "(32.2)"], ["Net earnings", "", "", "$224.6"], ["Net earnings attributable to Black Knight", "$108.8", "$168.5", ""], ["Shares used for diluted net earnings per share:", "", "", ""], ["Weighted average shares of common stock outstanding", "147.7", "147.6", "88.7"], ["Dilutive effect of unvested restricted shares of common", "", "", ""], ["stock", "0.9", "0.6", "0.6"], ["Weighted average shares of BKFS Class B common stock outstanding", "", "", "63.1"], ["Weighted average shares of common stock, diluted", "148.6", "148.2", "152.4"], ["Diluted net earnings per share", "$0.73", "$1.14", "$1.47"]]}

Equation:
0.74-2.06

Example 2 - The response is determined from the text or table
Here is a instruction detailing a task, accompanied by input and data providing additional context. Provide a suitable reply that effectively fulfills the inquiry.

Instruction:
What was the Additions based on tax positions related to current year in 2019 and 2018 respectively?

Input:
A reconciliation of the beginning and ending amount of unrecognized tax benefits is as follows: Interest and penalty charges, if any, related to uncertain tax positions are classified as income tax expense in the accompanying consolidated statements of operations. As of March 31, 2019 and 2018, the Company had immaterial accrued interest or penalties related to uncertain tax positions. The Company is subject to taxation in the United Kingdom and several foreign jurisdictions. As of March 31, 2019, the Company is

no longer subject to examination by taxing authorities in the United Kingdom for years prior to March 31, 2017. The significant foreign jurisdictions in which the Company operates are no longer subject to examination by taxing authorities for years prior to March 31, 2016. In addition, net operating loss carryforwards in certain jurisdictions may be subject to adjustments by taxing authorities in future years when they are utilized. The Company had approximately $24.9 million of unremitted foreign earnings as of March 31, 2019. Income taxes have been provided on approximately $10.0 million of the unremitted foreign earnings. Income taxes have not been provided on approximately $14.9 million of unremitted foreign earnings because they are considered to be indefinitely reinvested. The tax payable on the earnings that are indefinitely reinvested would be immaterial.

Data:
{"header": ["", "Year ended March 31,", ""], "rows": [["", "2019", "2018"], ["Beginning balance", "$6,164", "$4,931"], ["Additions based on tax positions related to current year", "164", "142"], ["Additions for tax positions of prior years", "231", "1,444"], ["Reductions due to change in foreign exchange rate ", "(301)", "(353)"], ["Expiration of statutes of limitation", "(165)", ""], ["Reductions due to settlements with tax authorities", "(77)", ""], ["Ending balance", "$6,016", "$6,164"]]}

Response:
164, 142

Example 3 - The response is an equation
Below is an instruction that describes a task, coupled with input and data providing additional context. Formulate an arithmetic equation to generate the answer.

Instruction:
What is the average value per share that Robert Andersen acquired on vesting?

Input:
Option Exercises and Stock Vested The table below sets forth information concerning the number of shares acquired on exercise of option awards and vesting of stock awards in 2019 and the value realized upon vesting by such officers. (1) Amounts realized from the vesting of stock awards are calculated by multiplying the number of shares that vested by the fair market value of a share of our common stock on the vesting date.

Data:
{"header": ["", "Option Awards", "", "Stock Awards", ""], "rows": [["Name", "Number of Shares Acquired on Exercise (#)", "Value Realized on Exercise ($)", "Number of Shares Acquired on Vesting (#)", "Value Realized on Vesting ($)"], ["Jon Kirchner", "", "", "153,090", "3,428,285"], ["Robert Andersen", "", "", "24,500", "578,806"], ["Paul Davis", "", "", "20,500", "482,680"], ["Murali Dharan", "", "", "15,000", "330,120"], ["Geir Skaaden", "", "", "21,100", "500,804"]]}

Equation:
578,806/24,500

C.2 PhraseBank

Example 1
Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:
Determine the sentiment of the following.

Input:
The plant will be fired with a combination of spruce bark, chipped logging residues or milled peat.

Response:
neutral

Example 2
Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:
Determine the sentiment of the following.

Input:
Operating profit improved by 27% to EUR 579.8mn from EUR 457.2mn in 2006.

Response:
positive

C.3 Wiki-SQL

Example 1
Below is an instruction that describes a task, coupled with contextual data. Compose an SQL script capable of being run on the data to generate the solution.

Instruction:
How many people watched at Glenferrie Oval?

Data:
{"header": ["Home team", "Home team score", "Away team", "Away team score", "Venue", "Crowd", "Date"], "rows": [["North Melbourne", "12.10 (82)", "South Melbourne", "11.14 (80)", "Arden Street Oval", "6,000", "4 August 1928"], ["Fitzroy", "13.12 (90)", "Footscray", "12.17 (89)", "Brunswick Street Oval", "12,000", "4 August 1928"], ["Richmond", "11.13 (79)", "Melbourne", "7.8 (50)", "Punt Road Oval", "26,000", "4 August 1928"], ["Geelong", "4.14 (38)", "Essendon", "12.10 (82)", "Corio Oval", "10,000", "4 August 1928"], ["Hawthorn", "9.9 (63)", "Collingwood", "17.18 (120)", "Glenferrie Oval", "5,000", "4 August 1928"], ["St Kilda", "13.15 (93)", "Carlton", "10.9 (69)", "Junction Oval", "31,000", "4 August 1928"]], "types": ["text", "text", "text", "text", "text", "real", "text"], "caption": "Round 15"}

SQL:
SELECT SUM([Crowd]) FROM data_table WHERE LOWER([Venue]) = LOWER('glenferrie oval')'

C.4 OTT-QA

Example 1
Here is a instruction detailing a task, accompanied by data providing additional context. Provide a suitable reply that effectively fulfills the inquiry.

Instruction:
How many kilometers is the airport from the Australian city known for housing the Towsers Huts?

Data:
{"header": ["Community", "Airport name", "Type", "ICAO", "IATA"], "rows": [["Albury", "Albury Airport", "Public", "YMAY", "ABX"], ["Armidale", "Armidale Airport", "Public", "YARM", "ARM"], ["Ballina", "Ballina Byron Gateway Airport", "Public", "YBNA", "BNK"], ["Balranald", "Balranald Airport", "Public", "YBRN", "BZD"], ["Bankstown , Sydney", "Bankstown Airport", "Airschool", "YSBK", "BWU"], ["Bathurst", "Bathurst Airport", "Public", "YBTH", "BHS"], ["Bourke", "Bourke Airport", "Public", "YBKE", "BRK"], ["Brewarrina", "Brewarrina Airport", "Public", "YBRW", "BWQ"], ["Broken Hill", "Broken Hill Airport", "Public", "YBHI", "BHQ"], ["Camden", "Camden Airport",

"Public", "YSCN", "CDU"], ["Cessnock", "Cessnock Airport", "Public", "YCNK", "CES"], ["Cobar", "Cobar Airport", "Public", "YCBA", "CAZ"], ["Coffs Harbour", "Coffs Harbour Airport", "Public", "YCFS", "CFS"], ["Collarenebri", "Collarenebri Airport", "Public", "YCBR", "CRB"], ["Condobolin", "Condobolin Airport", "Public", "YCDO", "CBX"], ["Coolah", "Coolah Airport", "Public", "YCAH", ""], ["Cooma", "Cooma - Polo Flat Airport", "Public", "YPFT", ""], ["Cooma", "Cooma - Snowy Mountains Airport", "Public", "YCOM", "OOM"], ["Coonabarabran", "Coonabarabran Airport", "Public", "YCBB", "COJ"], ["Coonamble", "Coonamble Airport", "Public", "YCNM", "CNB"]], "caption": "List of airports in New South Wales"}

Response:
3

C.5 Template choice

Example 1 - Arithmetic Template
Here is a instruction, input and data detailing a task. Which template is best suited to fulfil this inquiry.

Instruction:
What was the % change in gains recognized in other comprehensive income (loss), net of tax of $1, $11, and $4 from 2018 to 2019?

Input:
Cash Flow Hedge Gains (Losses) We recognized the following gains (losses) on foreign exchange contracts designated as cash flow hedges: We do not have any net derivative gains included in AOCI as of June 30, 2019 that will be reclassified into earnings within the following 12 months. No significant amounts of gains (losses) were reclassified from AOCI into earnings as a result of forecasted transactions that failed to occur during fiscal year 2019.

Data:
{"header": ["(In millions)", "", "", ""], "rows": [["Year Ended June 30,", "2019", "2018", "2017"], ["Effective Portion", "", "", ""], ["Gains recognized in other comprehensive income (loss), net of tax of $1, $11, and $4", "$ 159", "$ 219", "$ 328"], ["Gains reclassified from accumulated other comprehensive income (loss) into revenue", "341", "185", "555"], ["Amount Excluded from Effectiveness Assessment and Ineffective Portion", "", "", ""], ["Losses recognized in other income (expense), net", "(64)", "(255)", "(389)"]]}

Template:
arithmetic

Example 2 - Script Template
Here is a instruction and data detailing a task. Which template is best suited to fulfil this inquiry.

Instruction:
In what division was there a population density in km2 of 4,491.8 in 2011?

Data:
{"header": ["Administrative division", "Area (km) 2011**", "Population 2001 Census (Adjusted)", "Population 2011 Census (Adjusted)", "Population density (/km 2011)"], "rows": [["Dhaka District", "1,463.6", 9036647, 12517361, "8,552.4"], ["=> Savar Upazila", "282.11", 629695, 1442885, "5,114.6"], ["=> Keraniganj Upazila", "166.82", 649373, 824538, "4,942.68"], ["Narayanganj District", "684.37", 2300514, 3074078, "4,491.8"], ["=> Narayanganj Sadar Upazila", "100.74", 946205, 1381796, "13,716.5"], ["=> Bandar Upazila", "54.39", 267021, 327149, "6,014.8"], ["=> Rupganj Upazila", "176.48", 423135, 558192, "3,162.9"], ["Gazipur District", "1,806.36", 2143200, 3548115, "1,964.2"], ["=> Gazipur Sadar Upazila", "457.67", 925454, 1899575, "4,150.5"], ["=> Kaliakair Upazila", "314.13", 278967, 503976, "1,604.3"], ["Narsingdi District", "1,150.14", 1983499, 2314899, "2,012.7"], ["=>

Narsingdi Sadar Upazila", "213.43", 606474, 737362, "3,454.8"], ["=> Palash Upazila", "94.43", 198106, 221979, "2,350.7"]], "types": ["text", "text", "real", "real", "text"]}

Template:
script