SOURCE2SYNTH: SYNTHETIC DATA GENERATION AND CURATION GROUNDED IN REAL DATA SOURCES

Alisia Lupidi[1,2], Carlos Gemmell[1], Nicola Cancedda [1], Jane Dwivedi-Yu [1], Jason Weston [1], Jakob Foerster [2], Roberta Raileanu[1,3], Maria Lomeli[1]
[1]Meta, [2]Oxford University, [3]University College London

ABSTRACT

Synthetic data generation has recently emerged as a promising approach for enhancing the capabilities of large language models (LLMs) without the need for expensive human annotations. However, existing methods often generate data that can be low quality or contrived. In this paper, we introduce *Source2Synth*, a scalable approach for synthetic data generation and curation that is grounded in real-world data sources. Source2Synth takes as input a custom data source and produces synthetic data points with intermediate reasoning steps. Our method improves the dataset quality by discarding low-quality generations based on their answerability. We demonstrate the generality of this approach by applying it to two tasks that leverage two different types of sources: multi-hop question answering (MHQA), where we test complex reasoning abilities leveraging documents, and tabular question answering (TQA), where we test tool usage leveraging tables. Our method improves performance by 25.51% for TQA on WikiSQL and 22.57% for MHQA on HotpotQA compared to the fine-tuned baselines.

1 INTRODUCTION

Large Language Models (LLMs) (Devlin et al., 2019; Chowdhery et al., 2022; Brown et al., 2020; Vaswani et al., 2017) have risen to popularity due to their remarkable ability to digest and generate human-like text (Radford et al., 2018). However, it is difficult to unlock new capabilities for LLMs to solve more complex tasks due to the unavailability of task-specific data. Some examples of such complex tasks are multi-step reasoning, tool use and manipulating or processing structured data, among others. Enriching the data with human annotations collected for specific tasks is an expensive and time-consuming process (Touvron et al., 2023) which is subject to human-errors and bias.

In this paper, we propose the Source2Synth self-augmentation and self-improvement approach to produce *high quality synthetic data grounded in external real-world sources*. Basing the data generation process on real-world sources steers the examples to be more realistic, diverse, and factually correct. The self-improvement step via curation enables to filter out low quality data. We showcase our method on two challenging tasks that leverage different data sources: multi-hop question-answering (based on documents to test multi-step reasoning and information extraction), and tabular question answering (based on tables, testing tool-use via SQL). In both cases, models trained with Source2Synth's pipeline achieve improved performance without relying on human annotations, resulting in a scalable data generation method for complex tasks. To summarize, our key contributions are: 1) a new scalable method for generating synthetic data grounded in a real data source for a given task, and 2) a curation method based on filtering and imputation which yields higher quality data and improved task performance.

2 RELATED WORK

Synthetic Data Generation using LLMs A number of works propose leveraging language models to generate synthetic datasets. Some rely on knowledge-probing, by generating a continuation or predicts missing words in a close-style template (Schick & Schütze, 2020; Schick & Schütze, 2021; Petroni et al., 2019; Jiang et al., 2019), while others improve the quality of synthetic data by using different model-based or human filtering techniques (Schick & Schütze, 2021; Liu et al., 2022; Li et al., 2024; Thoppilan et al., 2022). Our method improves the quality of the synthetic data by leveraging the LLM itself bypassing human-in-the-loop steps, thus being cheaper and more scalable. Our seed selection topic is automated, and we leverage real data as a starting point, which steers the examples to be more realistic, diverse, and factually correct. Please see Appendix C for a review of works that leverage real-world data sources and a comparison to our method.

Teaching LLMs to Use Tools Enabling LLMs with tool-use extends their abilities to manipulating structured data, retrieving information from external sources, or interacting with APIs. Various works augment LLMs with general tools or API calls (Parisi et al., 2022; Schick et al., 2023; Tang et al., 2023), possibly interleaving reasoning steps with API calls (Gao et al., 2023; Cai et al., 2024; Paranjape et al., 2023). Finally, some works investigate the use of unseen tools at test time (Paranjape et al., 2023; Mekala et al., 2024). See Mialon et al. (2023) and Qin et al. (2023) for an in-depth review of augmented language models research. In the above approaches tool usage is restricted to inputs that are strings or numbers. However, using structured data (like tables and graphs) during post-training can be useful to enhance the LLM's capabilities in complex tasks. A particular tool of interest is SQL since it enables aggregating information from tabular data, see Appendix C for more information on using SQL as a tool in LLMs.

3 Method

Source2Synth provides a way to generate high-quality synthetic datasets for a given application leveraging different types of real-world data sources by self-augmentation and self-improvement. The generated synthetic examples can then be used as augmented step-by-step examples for fine-tuning the LLM. Source2Synth is composed of three stages: *Dataset Generation*, *Dataset Curation*, and *Model Fine-tuning*.

Dataset Generation

Data source selection We first select a data source. The source can be an already existing dataset re-purposed for the task of interest, a collection of existing data points, or structured information (e.g. graphs, tables). There is no need for human annotations on the entries, as Source2Synth enriches it with extra instructions by self-augmentation.

Seed To create a synthetic example, we first generate a *seed* topic. The seed is chosen at random from the source data. The seeds anchors the creation of the entry, making it consistent through the succesive steps of the generation process.

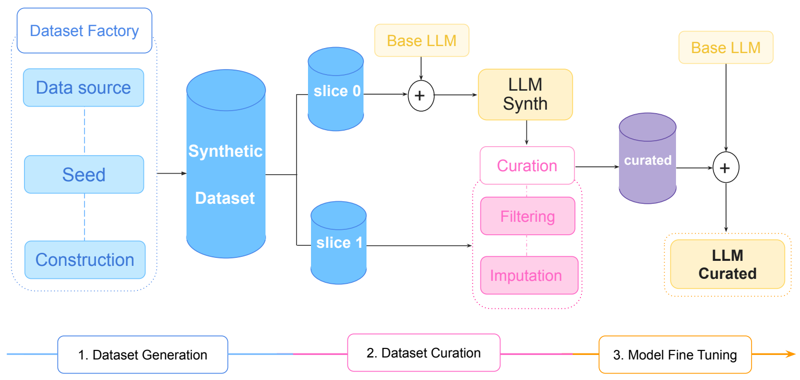


Figure 1: **Source2Synth Method.** During *Dataset Generation*, we choose a data source to build the dataset. For each example, we select a seed topic to condition the generation on, and use the data source and seed together to construct the example. The resulting synthetic dataset is sliced in two: slice 0, used to fine-tune an intermediate version of the LLM (*LLMSynth*), which is then used to curate slice 1 through filtering and/or imputation during *Dataset Curation* step. The curated dataset is of higher quality and aligned with the user's design. At *Model Fine-tuning* stage, the final LLM (*LLMCurated*) is trained on the curated synthetic dataset.

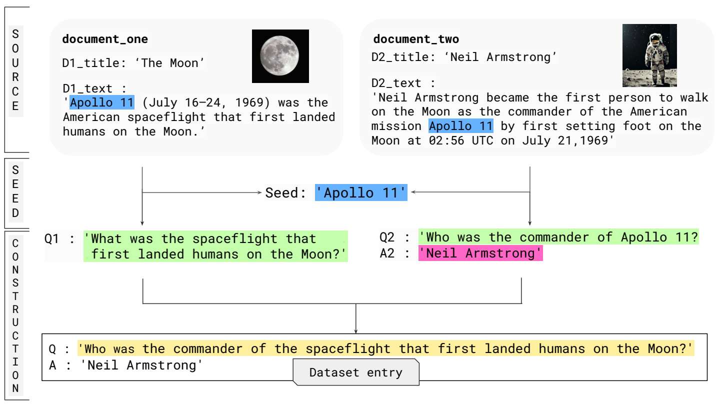
Dataset construction In order to leverage complex tasks with LLMs, typically we can resort to Chain-Of-Thought (Wei et al., 2022) prompting. Analogously, in Source2Synth we leverage the seed to build synthetic step-by-step data, decomposing the generation into intermediate steps. This reasoning chain augmentation can be used as supervision (for reasoning or for learning tool-use) by providing it as the target in the synthetically generated training examples.

Dataset Curation
During curation, the dataset is split in two halves: the first half is used to fine-tune a LLM (*LLMSynth*), *LLMSynth* is then used to self-improve the quality of the second slice of the dataset by performing imputation plus a filtering step. After these steps, we obtain the final curated dataset (in purple in Figure 1).
Data filtering The finetuned model *LLMSynth* is used to predict the output of the given synthetic example using k tries. If the output cannot be predicted at least once, it is assumed the example is low quality and is not included in the final curated dataset.
Data Imputation We also consider an imputation process, which involves blanking parts of the augmented data points and using the LLM to fill in the blanks, to replace those fields.

Model Fine-tuning
We fine-tune a pretrained or instruction-tuned version of the LLM using the Source2Synth synthetic dataset. We use our dataset for supervised training of both the reasoning chain and the final answer. The resulting *LLMCurated* model is then equipped with the relevant capability for the task of interest.

4 APPLICATIONS

The general pipeline described above can be used to produce custom synthetic examples tailored for the task at hand (like teaching LLMs new skills). We evaluate Source2Synth on two challenging tasks: multi-hop question answering (leveraging documents as sources, to test reasoning), and tabular question answering(leveraging tables, to test tool-use).



D1_title: 'The Moon'

D1_text : 'Apollo 11 (July 16–24, 1969) was the American spaceflight that first landed humans on the Moon.'

D2_title: 'Neil Armstrong'

D2_text : 'Neil Armstrong became the first person to walk on the Moon as the commander of the American mission Apollo 11 by first setting foot on the Moon at 02:56 UTC on July 21,1969'

Q1 : 'What was the spaceflight that first landed humans on the Moon?'

Q2 : 'Who was the commander of Apollo 11?
A2 : 'Neil Armstrong'

Q : 'Who was the commander of the spaceflight that first landed humans on the Moon?'
A : 'Neil Armstrong'

Figure 2: **MHQA synthetic data generation process:**. randomly pick one article D_1, e.g. "The Moon" article. Next, at *Seed* stage, an entity E is retrieved from D_1 - e.g. "Apollo 11". Then, we sample from the pool of related documents of D_1 where E is present e.g. we select D_2 titled "Neil Armstrong". A question Q_1 is generated from D_1, with the constraint that the answer A_1 is the entity E. A second question Q_2 is generated from D_2, with the constraint that its main topic is the entity E. We then prompt an LLM to merge the two questions into a multi-hop one Q, based on their common entity. The training example comprises of Q, A, the sub-questions, reasoning chain, and the entity.

4.1 MULTI-HOP QUESTION ANSWERING

For the multi-hop question answering task (MHQA), we generate a synthetic dataset consisting of a multi-hop question Q with its answer A, the decomposition into sub-questions with answers and the relative reasoning chain, plus the entity that links the sub questions. See Figure 2 for an overview of the procedure and Figure 4 - Right for an example response from the model finetuned with the Source2Synth approach.

Dataset Generation
Data source selection We use English Wikipedia (Wikipedia contributors, 2004) as the data source for MHQA, since it contains articles in natural language and additional meta-information like links to related articles. Firstly, we randomly select an initial article, denoted as D_1, among all available Wikipedia articles. For each D_1 we collect $n \geq 2$ related articles.

Seed An MHQA seed topic corresponds to an entity E retrieved from D_1. The seed topic in MHQA is also used as the "hop" in the multi-hop question Q that we aim to generate, since E links the $n = 2$ sub-questions that compose Q. In Figure 2, we sample the article D_1 ="The Moon" at random and sample a corresponding entity, E ="Apollo 11" in this case. Then, we pick D_2 ="Neil Armstrong" from the pool of related articles, since it contains a paragraph where the entity "Apollo 11" is included.

Dataset construction
We prompt an instruction-tuned language model to generate two questions: a question Q_1 based on D_1, whose answer is the selected entity E; and a second question Q_2, based on D_2 such that its main topic is E. See Figures 16 and 17 for the prompts. In Figure 2, Q_1 = "What was the spaceflight that first landed humans on the Moon?", the hop is E = "Apollo 11" and Q_2 = "Who was the commander of Apollo 11?". We then prompt the LLM to merge the two questions, in order to generate the final two-hop question Q by using the entity as a conceptual link (hop). The prompt is given in Figure 15.

Dataset Curation
Data filtering We check if the predicted answer matches the answer deterministically extracted in the synthetically generated example, and if after $k = 3$ tries tries the LLM has not supplied the correct answer we filter out the entry entirely. See Figure 4 - Left for an example of model inference.

Data Imputation For MHQA, we blank Q_1 and provide the LLM with Q, Q_2, E, and the relative doc sample D_1 as context when asking it to reconstruct Q_1. The new candidate Q'_1 for Q_1 is then assessed: if A' (the answer to the new multi-hop question Q' resulting from assembling Q'_1 and Q_2) matches A (the original answer to Q) then we keep the example. We find that asking the model to reconstruct parts of the multi-hop question in-context results in a more natural and cohesive question, thus removing some of the unnaturalness of the text that can occur from automatically generated and merged examples (see Appendix E.4 for a study on the advantages of imputation). In total, the curation step removes around 13% of the questions originally generated.

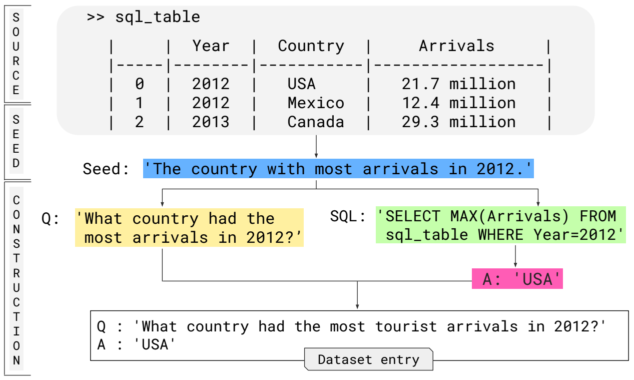
4.2 Tabular question answering

In Tabular question answering (TQA), we generate a question-answer dataset where each question is based on a table from the data source. Generated training examples are hence augmented with annotations built from automatically-generated interesting facts retrieved from the table.

Dataset Generation

Data source selection In the TQA case, we use 4k unlabeled tables from the WikiSQL (Zhong et al., 2017) training dataset as sources.

Seed For each table, we prompt an instruction-tuned language model to generate a statement based on the table. This statement is the seed topic for the generation and is a pertinent interesting fact or set of observations in natural language that can be derived from the table. The prompt is given in Figure 11.



```
>> sql_table
    |     |  Year  |  Country  |     Arrivals      |
    |-----|--------|-----------|-------------------|
    |  0  |  2012  |   USA     |   21.7 million    |
    |  1  |  2012  |   Mexico  |   12.4 million    |
    |  2  |  2013  |   Canada  |   29.3 million    |
```

```
Seed: 'The country with most arrivals in 2012.'
Q: 'What country had the most arrivals in 2012?'
SQL: 'SELECT MAX(Arrivals) FROM sql_table WHERE Year=2012'
A: 'USA'
Q : 'What country had the most tourist arrivals in 2012?'
A : 'USA'
```

Figure 3: **TQA synthetic data generation process.** Firstly, generate the seed: a fact based on the table (in blue). Given the seed and table, an *SQL query* is generated (in green) as well as its translation into natural language *Q*. Then, the SQL is executed to obtain the answer **A**.

Dataset construction We next generate an SQL-statement by zero-shot prompting the LLM: we provide the table and the seed (factual statement) as context, see Figure 12 for the prompt. Given the produced SQL statement, it is then executed using the Python library *sqlite3*[1] to obtain an SQL answer formatted as a table. If the generated statement is invalid, we discard it and re-generate. We generate a total of 10k SQL statements based on the source tables. We execute the statements to filter for validity (i.e. discarding non-executable SQL statements) and the final dataset size is 8k per slice.

Dataset Curation
Data filtering We check if the predicted answer of *LLMSynth* fine-tuned on slice 0 matches the answer in the synthetically generated example, and if after $k = 3$ tries the model has not supplied the correct

[1]https://www.sqlite.org

answer we filter out the entry entirely. See Figure 4 - Right for an example of model inference. In TQA, the curation process consists only of the filtering step. After curation, we keep 27% of the original examples in slice 1.

5 Experimental Setup

We test our method on two tasks that leverage two different data-types as sources: *tabular question answering*, which uses tables, and *multi-hop question answering*, which uses documents. We use Source2Synth to generate and curate a high-quality data set suitable for fine-tuning and compare our method with a number of baselines.see Appendix D for more details on the data source, metrics, and model used in both applications.

Multi-Hop QA Setup - Baselines
We compare Source2Synth to the following baselines (for all the listed models, we use two prompt templates, a zero-shot and a three-shot CoT, see Figure 14, Appendix G):
Instruction-tuned LLM: we prompt LLama-2 70B-Chat for the task.
Fine-tuned LLM (HotpotQA only): we fine-tune Llama-2 70B-Chat model with 500 examples from the HPQA training split.
LLMSynth (Synthetic dataset only): we fine-tune LLama-2 70B-Chat model with 1250 synthetic examples from Slice 0 (see Figure 1), *without* the data curation step.
LLMSynth-datamix (Synthetic and HotpotQA): we fine-tune LLama-2 70B-Chat with the uncurated synthetic data in addition to the 500 HPQA examples.

Tabular QA Setup - Baselines
We compare the performance of our Source2Synth method against a variety of baselines consisting of prompting the Starchat-beta instruction-tuned language model:
Zero-shot Table QA (Figure 7): zero-shot prompt with task instruction, table and question.
One-Shot No Context QA (Figure 8): one-shot example with a question and answer prompt with task instruction and the actual question to answer.
One-Shot Table QA (Figure 9): prompt including the table for the one-shot example and the question to answer. We use one-shot only due to context-length limitations and the large size of tables.
One-shot Table+SQL QA (Figure 10): the prompt includes an example containing the table and question, and an instruction suggesting that the model can leverage an SQL tool. We then execute the predicted SQL to obtain the answer.
LLMSynth: Fine-tune the model with synthetic data *without* applying the data curation step.

6 Results

Multi-Hop question answering

We report the experimental results in Table 1. We include the following baselines and use a zero-shot and three-shot promps, see Figure 14): 1) prompting a standard instruction-tuned LLM (first row), 2) a *fine-tuned LLM* using only the HotpotQA 500 examples from the train split (second row), and 3) *LLMSynth* which only uses the uncurated synthetic data for fine-tuning (third row), and lastly *LLMSynth-datamix* that is fine-tuned on both HotpotQA and the uncurated synthetic dataset (fourth-row). All fine-tuned methods outperform the first baseline where we prompt the instruction-tuned model. Using only synthetic data or only HotpotQA data for fine-tuning demonstrates worse performance than when combined, whether the synthetic data is curated (*LLMCurated*, fifth row), or not (*LLMSynth*, third row). All models where we use the

Table 1: **Evaluation of Source2Synth on Multi-hop question answering.** The models shown are fine-tuned with 500 entries from HotpotQA and/or 1250 entries from the Source2Synth Synthetic data. Using Source2Synth curated synthetic data in combination with HotpotQA (*LLMCurated-datamix*, last row) works best.

Method	**0-shot**	**3-shot CoT**
Instruction-tuned LLM (LLama 2 70B-Chat)	40.45%	44.13%
Fine-tuned LLM (HotpotQA data only)	53.22%	58.40%
LLMSynth (Synthetic data only)	52.31%	56.70%
LLMSynth-datamix (Synthetic and HotpotQA)	57.46%	62.73%
LLMCurated (Synthetic data only)	64.07%	64.68%
***LLMCurated-datamix* (Synthetic and HotpotQA)**	**65.23%**	**66.05%**

full Source2Synth pipeline we see further performance improvements (*LLMCurated*, *LLMCurated-datamix*, fifth and sixth rows) vs not curating the data (*LLMSynth*, *LLMSynth-datamix*, third and fourth row).

Source2Synth on grounded synthetic data points only (without HotpotQA training examples) We fine-tune Llama2-70B-Chat only on grounded synthetic data, to simulate the no-data regime (i.e. the data points come solely from Source2Synth's pipeline starting from a real-world source). In Table 1 we see that fine-tuning without those additional 500 entries from HotpotQA minimally hinders performance. Furthermore, we evaluate *LLMCurated* on the difficulty splits in Table 3. While the model achieves comparable performance on bridge questions, the model shows slightly worse performance on comparison questions (Table 3, row 3).

Other experiments 1) In Appendix E.1 we break down the model's performance with respect to the difficulty levels of the questions, and show that the model not only improved in multi-hop QA, but by fine-tuning with data from our pipeline it gained better reasoning abilities on harder questions; 2) In Appendix E.2 we present an analysis of Source2Synth's performance (both *LLMSynth* and *LLMCurated*) with respect to the training dataset size.

Other ablations 1) In Appendix E.3, we apply the whole pipeline using an ungrounded synthetic dataset to showcase the benefit of using real-world data as a source to generate the synthetic dataset. 2) In Appendix E.4 we measure the naturalness of generated MHQA questions by measuring perplexity before and after imputation. 3) In Appendix E.5, we study how Source2Synth performs when using a smaller model (Llama3-8B instruct) and/or on synthetic data not generated in a monolithic fashion.

Tabular question answering

We report the experimental results for TQA in Table 2. We see that providing no context about the table when prompting the instruction-tuned StarChat language model has very poor performance (first row). This is expected, since questions in WikiSQL require information contained in the table to answer, while the model does not have any other information except for the general knowledge stored in its parameters. However, even if we pass the table as part of the prompt, the performance does not improve much due to its difficulties to digest structured data (second row). While passing an example of table usage in a one-shot fashion (third row) improves the soft-EM metric, the EM metric is still very low (model gathers the correct info but does not understand it correctly).

The performance increases once we provide a one-shot example containing the relevant table and SQL query (fourth row). This means that the model's ability to leverage the SQL tool improves performance markedly. When we use the Source2Synth curated data to fine-tune the StarChat model (last row), we observe a significant increase in performance: indeed, our full method performs significantly better than fine-tuning the StarChat language model using synthetic data without curation, *LLMSynth* (second to last row), although that still outperforms the other baselines by a large margin as well, indicating the utility of our Source2Synth synthetic data generation scheme.

Table 2: **Tabular question answering.** The models are fine-tuned using Source2Synth curated synthetic data only. Performance comparison on the WikiSQL evaluation dataset.

Method	**Exact Match**	**Soft-EM**
One-Shot No Context QA (Starchat-beta LLM)	0.25%	16.22%
Zero-shot Table QA (Starchat-beta LLM)	1.83%	20.07%
One-Shot Table QA (Starchat-beta LLM)	2.03%	31.06%
One-shot Table+SQL QA (Starchat-beta LLM)	12.30%	34.13%
LLMSynth (Synthetic data only)	23.86%	34.21%
***LLMCurated* (Synthetic data only)**	**34.50%**	**42.80%**

7 CONCLUSIONS

We introduce Source2Synth, a new method for generating and curating high-quality synthetic data grounded in real data sources. We demonstrate its utility on two tasks, that leverage two different data source types, and pose significant challenges for LLMs: multi-hop reasoning and tabular question answering with SQL. We see our method as a first step towards building high-quality automatic data generation methods without human input.

REFERENCES

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers, 2024. URL https://arxiv.org/abs/2305.17126.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. Pal: Program-aided language models, 2023. URL https://arxiv.org/abs/2211.10435.

Carlos Gemmell and Jeffrey Dalton. Generate, transform, answer: Question specific tool synthesis for tabular data, 2023. URL https://arxiv.org/abs/2303.10138.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. TaPas: Weakly supervised table parsing via pre-training. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4320–4333, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.398. URL https://aclanthology.org/2020.acl-main.398.

Zhengbao Jiang, Frank F. Xu, J. Araki, and Graham Neubig. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438, 2019. URL https://api.semanticscholar.org/CorpusID:208513249.

Jinyang Li, Binyuan Hui, Ge Qu, Binhua Li, Jiaxi Yang, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls, 2023a.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder: may the source be with you!, 2023b. URL https://arxiv.org/abs/2305.06161.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason Weston, and Mike Lewis. Self-alignment with instruction backtranslation, 2024. URL https://arxiv.org/abs/2308.06259.

Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. WANLI: Worker and AI collaboration for natural language inference dataset creation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 6826–6847, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.508. URL https://aclanthology.org/2022.findings-emnlp.508.

Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, and Andrew M. Dai. Best practices and lessons learned on synthetic data, 2024. URL https://arxiv.org/abs/2404.07503.

Dheeraj Mekala, Jason Weston, Jack Lanchantin, Roberta Raileanu, Maria Lomeli, Jingbo Shang, and Jane Dwivedi-Yu. Toolverifier: Generalization to new tools via self-verification, 2024. URL https://arxiv.org/abs/2402.14158.

Grégoire Mialon, Roberto Dessi, Maria Lomeli, Christoforos Nalmpantis, Ramakanth Pasunuru, Roberta Raileanu, Baptiste Roziere, Timo Schick, Jane Dwivedi-Yu, Asli Celikyilmaz, Edouard Grave, Yann LeCun, and Thomas Scialom. Augmented language models: a survey. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=jh7wH2AzKK. Survey Certification.

Thao Nguyen, Jeffrey Li, Sewoong Oh, Ludwig Schmidt, Jason Weston, Luke Zettlemoyer, and Xian Li. Better alignment with instruction back-and-forth translation, 2024. URL https://arxiv.org/abs/2408.04614.

Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. Art: Automatic multi-step reasoning and tool-use for large language models, 2023. URL https://arxiv.org/abs/2303.09014.

Aaron Parisi, Yao Zhao, and Noah Fiedel. Talm: Tool augmented language models, 2022. URL https://arxiv.org/abs/2205.12255.

Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In Chengqing Zong and Michael Strube (eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1470–1480, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1142. URL https://aclanthology.org/P15-1142.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. Language models as knowledge bases? In *Conference on Empirical Methods in Natural Language Processing*, 2019. URL https://api.semanticscholar.org/CorpusID:202539551.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. Tool learning with foundation models, 2023. URL https://arxiv.org/abs/2304.08354.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. 2018.

Timo Schick and Hinrich Schütze. Few-shot text generation with natural language instructions. In *Conference on Empirical Methods in Natural Language Processing*, 2020. URL https://api.semanticscholar.org/CorpusID:238260199.

Timo Schick and Hinrich Schütze. Generating datasets with pretrained language models. *ArXiv*, abs/2104.07540, 2021. URL https://api.semanticscholar.org/CorpusID:233241169.

Timo Schick and Hinrich Schütze. Exploiting cloze questions for few shot text classification and natural language inference, 2021. URL https://arxiv.org/abs/2001.07676.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=Yacmpz84TH.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *ArXiv*, abs/2306.05301, 2023. URL https://api.semanticscholar.org/CorpusID:259108190.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. Lamda: Language models for dialog applications, 2022. URL https://arxiv.org/abs/2201.08239.

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, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:13756489.

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

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with oss-instruct, 2024. URL https://arxiv.org/abs/2312.02120.

Wikipedia contributors. Plagiarism — Wikipedia, the free encyclopedia, 2004. URL https://en.wikipedia.org/w/index.php?title=Plagiarism&oldid=5139350. [Online; accessed 22-July-2004].

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing*, 2018. URL https://api.semanticscholar.org/CorpusID:52822214.

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

Ingo Ziegler, Abdullatif Köksal, Desmond Elliott, and Hinrich Schutze. Craft your dataset: Task-specific synthetic dataset generation through corpus retrieval and augmentation. *ArXiv*, abs/2409.02098, 2024. URL https://api.semanticscholar.org/CorpusID:272367350.

A Limitations

In this paper, our applications use a single seed or table per query to derive questions. However, Source2Synth can be extended to more complex scenarios e.g. multiple hops or queries that require multi-table tool-use. This can be done by looping the dataset generation steps and feeding the result of the previous step as input to the next one. Our method could also be improved with more clever sampling techniques. We consider this to be an interesting avenue of future research.

B Ethics Statement

Although our work does not directly deal with private or sensitive data and we showcase applications starting on public sources, it can be used to create new datasets and/or stronger model starting from private data - for example, using Source2Synth to create a QA dataset on medical data. We believe that while this is of help to the community because it provides a way to self-generate synthetic data for an ad-hoc application / private source data, we encourage the user to be mindful of the context of deployment and take necessary precautions to preserve privacy.

C Extended related work

Using real-world sources Similarly, some recent works leverage real-world data in document form from the web to construct high-quality synthetic data (Nguyen et al., 2024; Ziegler et al., 2024) or open-source code snippets to generate diverse instruction data for code generation (Wei et al., 2024; Dubey et al., 2024). See Liu et al. (2024) for an overview of synthetic data research. Our general framework can be applied using data sources of different types. In contrast to these approaches, we do not require a back-translation approach or initial fine-tuning to generate the seed.

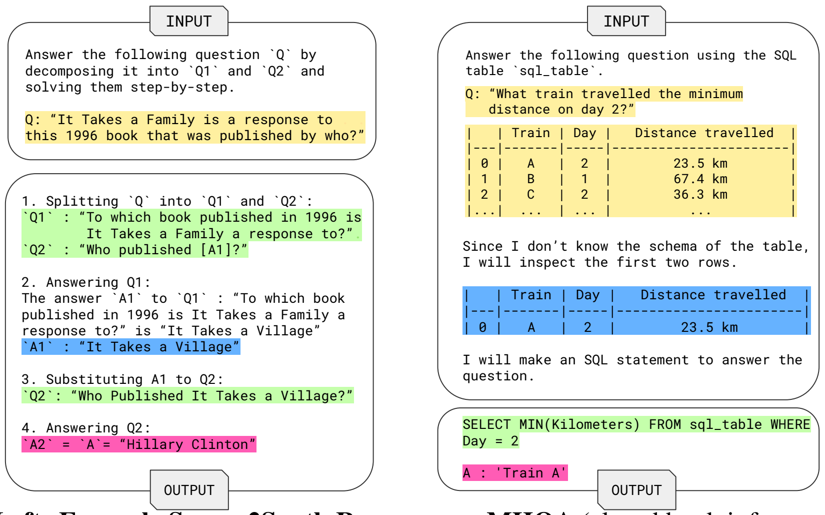
Teaching LLMs to use SQL In the above approaches usually tool usage is restricted to inputs that are strings or numbers. However, using structured data (like tables and graphs) during post-training can be useful to enhance the LLM's capabilities in complex tasks. A particular tool of interest is SQL since it enables aggregating information from tabular data. A variety of benchmarks (Pasupat & Liang, 2015) have been proposed to assess LLMs' abilities to generate SQL and their performance on tabular-based question answering with SQL (Li et al., 2023a; Zhong et al., 2017). Alternatively, other works studied how LLMs directly handle tabular data (Herzig et al., 2020; Gemmell & Dalton, 2023).

C.1 Inference output - illustration

D More Details on the Experimental Setup

D.1 Multi-hop QA

Data We evaluate Source2Synth on MHQA using the HotpotQA (Yang et al., 2018) (HPQA) benchmark: it contains 113,000 examples of multi-hop QA pairs based on Wikipedia, split in train, test, and validation sets. Yang et al. (2018) recommend the FullWiki setting to test multi-hop reasoning abilities of models. We use this setup in our evaluation with the test set. Each entry in HotpotQA is constructed such that: 1) each question requires finding and reasoning over multiple supporting documents in order to answer; 2) each entry provides sentence-level supporting facts for strong supervision and explain-ability of the prediction; 3) each question can be classified as either a comparison or bridge question. A comparison question entails comparing the same concept between n objects (e.g. "Who is the tallest student in class?"), while a bridge question builds on a logical and/or causal link and requires deriving statements to get to the answer (e.g. "What is the height of the student that topped the entry exam?" - this requires first identifying the student that topped the exam). The hop length is the number of comparison objects, for comparison questions, or the number of links, for bridge questions. We chose $n = 2$ to be consistent with HotpotQA . The test set consists of $7,405$ entries, split evenly between bridge and comparison questions. We only generate synthetic data for bridge questions, since they pose a bigger challenge to current LLMs and to counterbalance this disparity, we include 500 comparison questions from HotpotQA's training dataset in our fine-tuning dataset.



```
Answer the following question `Q` by
decomposing it into `Q1` and `Q2` and
solving them step-by-step.

Q: “It Takes a Family is a response to
this 1996 book that was published by who?”
```

```
1. Splitting `Q` into `Q1` and `Q2`:
`Q1` : “To which book published in 1996 is
        It Takes a Family a response to?”
`Q2` : “Who published [A1]?”

2. Answering Q1:
The answer `A1` to `Q1` : “To which book
published in 1996 is It Takes a Family a
response to?” is “It Takes a Village”
`A1` : “It Takes a Village”

3. Substituting A1 to Q2:
`Q2`: “Who Published It Takes a Village?”

4. Answering Q2:
`A2` = `A`= “Hillary Clinton”
```

```
Answer the following question using the SQL
table `sql_table`.
Q: “What train travelled the minimum
   distance on day 2?”
|   | Train | Day |   Distance travelled   |
|---|-------|-----|------------------------|
| 0 |   A   |  2  |        23.5 km         |
| 1 |   B   |  1  |        67.4 km         |
| 2 |   C   |  2  |        36.3 km         |
|...|  ...  | ... |          ...           |

Since I don’t know the schema of the table,
I will inspect the first two rows.

|   | Train | Day |   Distance travelled   |
|---|-------|-----|------------------------|
| 0 |   A   |  2  |        23.5 km         |

I will make an SQL statement to answer the
question.
```

```
SELECT MIN(Kilometers) FROM sql_table WHERE
Day = 2

A : 'Train A'
```

Figure 4: **Left: Example Source2Synth Response on MHQA** (closed book inference). Model's response (reasoning steps and answer) to a multi-hop input question (yellow). The colours highlight the generation of the augmented entries: the decomposition into sub questions, in green, the seed, in blue, and the final answer, in pink.
Right: Example Source2Synth Response on TQA. Model's response (SQL call and final answer) to the tabular input question (yellow). The coloured parts highlight the generation of the corresponding augmented entries: SQL, in green, the seed, in blue, and the final answer, in pink.

Metrics We measure the performance using soft exact match (soft-EM) as the metric. Soft-EM is 1 if the generated output contains the golden answer and 0 otherwise.

Model In MHQA experiments we use the Llama-2 70B-Chat LLM and we fine-tune Source2Synth and various other baseline methods starting from this model. Source2Synth is trained with 1250 synthetic examples, unless noted otherwise, in addition to the 500 HotpotQA examples above. The 1250 synthetic examples are generated starting from a collection of 50 randomly selected Wikipedia articles.

D.2 Tabular QA

Data We evaluate Source2Synth on TQA using the WikiSQL (Zhong et al., 2017) benchmark: it consists of a corpus of 80,654 hand-annotated examples of natural language questions, SQL queries, and tables created from 24,241 tables extracted from Wikipedia. The validation split contains 7,857 examples after removing non-executable SQL tables, see Appendix F for more details.

Metrics We measure performance using the exact match (EM) and the soft-EM metrics. The EM metric equals 1 if the golden answer is equal to the generated answer and 0 otherwise.

Model For TQA, we use the Starchat-beta language model (Li et al., 2023b) from Huggingface as the initial language model (batch size 32, 100 steps, lr 0.0001, linear warm-up). The Starchat model is an instruction-tuned LLM with 16 billion parameters trained to act as a helpful coding assistant. This model is a fine-tuned version of StarCoder (Li et al., 2023b), a LLM pre-trained on a large code corpus, which contains SQL statements, and fine-tuned on 35B Python tokens.

E Further experiments

E.1 Analysis of performance on different question types and levels of difficulty

In Table 3, we study the capabilities of our model by analysing the performance of LLM-Curated-1250 with a particular focus on the type and difficulty of the questions: hard/medium/easy bridge and comparison questions. We compare the performance of the base model, the model fine-tuned on HotpotQA only, and the model finetuned using Source2Synth, according to the difficulty level, provided in the HotpotQA train dataset. We also subdivide the results according to the type of question (bridge vs. comparison).

We observe that Source2Synth performs better across all types of questions and difficulties, with an average overall gain of 12.4% on the base LLM and a 7.5% gain compared to the LLM fine-tuned on HotpotQA. In particular, by applying our method, the resulting model is able to achieve +16.8% and +16.5% on hard bridge and comparison questions respectively, when comparing to the baseline. Furthermore, it is interesting to see substantial improvement on comparison-type questions, despite not explicitly targeting those during synthetic generation. Hard questions pose a greater challenge to the reasoning abilities of LLMs and these results introduce Source2Synth as a possible method for further improvement.

Table 3: **Analysis of MHQA bridge and comparison questions with respect to level of difficulty.** We evaluate models on the full HPQA train dataset (where questions are labelled with easy, medium and hard). Source2Synth outperforms the baseline and the fine-tuned on HotpotQA model, yielding a LLM capable of handling hard questions of both types.

	Bridge			*Comparison*		
Model	**Hard**	**Medium**	**Easy**	**Hard**	**Medium**	**Easy**
Llama2-70B-Chat	14.5%	27.2%	30.1%	66.6%	71.3%	73.2%
Fine-tuned LLM (HotpotQA data only)	20.1%	29.8%	34.3%	74.5%	78.3%	82.1%
LLMCurated-1250 (Synthetic data only)	27.6%	32.3%	36.2%	79.1%	82.3%	88%
***LLMCurated-datamix*-1250 (Synthetic and HotpotQA)**	**31.3%**	**35.6%**	**39.7%**	**83.1%**	**85.7%**	**87.8%**

E.2 Scaling performance

Source2Synth can be leveraged when the amount of available data is low. In Figure 5, we study how performance changes when adding more synthetic data in the fine-tuning data mix - which already includes 500 samples from the HPQA train split. We perform the analysis on both *LLMSynth-datamix* and *LLMCurated-datamix* to showcase the impact of the curation technique. In both cases and uniformly over all data mix sizes, we see that applying the Source2Synth pipeline results in a stronger model. For the *LLMSynth-datamix* model (fine-tuned on uncurated samples), providing more synthetic examples leads to a steady improvement in performance across all data sizes, for both zero-shot and three-shot prompting variants. The *LLMCurated-datamix* model follows a similar trend, consistently outperforming the uncurated version of the model, for all data mix set sizes. Overall, we observe that using our synthetic data generation pipeline to construct more synthetic data brings further performance gains.

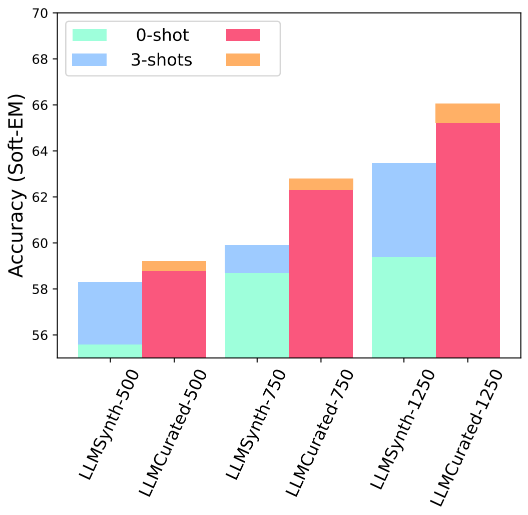


Figure 5: **Synthetic Data scaling performance.** Source2Synth's performance with respect to MHQA's data mix size, before and after curation. Sample sizes after curation: 7% for 500, 8% for 750, 11% for 1250.

E.3 Applying Source2Synth on Ungrounded data only

We deploy Source2Synth on an ungrounded synthetic dataset, consisting of 1250 synthetic data points (no real-world source) for fine-tuning plus 500 entries from HotpotQA. To ensure diversity in the generation, we ask the model to generate a question based on two topics A and B picked from the following list: [“Moon”, “Ocean”, “Water”, “Wolf”, “Tides”, “Day”, “Light”, “Apple”, “United States”, “Europe”, “Roman Empire”, “Chocolate”, “Environment”, “India”, “Strawberries”, “Physics”, “Pen”, “Sugar”, “History”, “Jelly”, “Mug”, “Cat”, “Lion”, “Flower”, “Purple”, “Red”, “Stars”, “Electricity”, “Paper”, “Snow”, “Mount Everest”, “Table”, “Friendship”, “Book”, “Laptop”, “Phone”, “Mushroom”, “Hat”, “Coffee”, “Pasta”, “Island”, “Volcano”, “Storm”, “Key”, “Candle”, “Asia”, “Desert”, “Tree”, “River”]

Model	Accuracy
Base LLM (Llama3-8B)	57.80%
LLMSynth	60.45%
LLMCurated	66.37%

Model	Accuracy
Base LLM (LLama2-70B)	40.45%
LLMSynth	51.90%
LLMCurated	59.70%

The resulting 2-hop questions are of bridge type and have lower perplexity than those generated starting from a grounded source. The model trained on ungrounded samples performs worse than the one trained on grounded ones. Since they are generated by the model without any other seed/input, there are repeating patterns in the structure of the questions generated, which we hypothesize hinders generalization. For example, many of the synthetically ungrounded questions follow this structure: ’What / Who [Q1] and / or What [Q2]?’ ” (i.e. "What is the name of the ancient mythological figure that is often depicted as being able to transform into a wolf, and is also associated with the full moon that occurs in March, which is also known as the Worm Moon?"). We test this setting on Llama2 and Llama3 using the 0-shot prompt.

E.4 On the impact of imputation

We studied the perplexity of questions before and after imputation for 1) synthetic data generated from a grounded source (like Wikipedia) and 2) for synthetic ungrounded data. In both cases, the imputation step lowers perplexity and reduces the unnaturalness of the question. In Table 4 we report

Comparing Q pre- and post- imputation

Before imputation:
Q: "What pet did the poet and father of mathematician Ada Lovelace had when he was a student at Trinity out of resentment for rules forbidding pet dogs like his beloved Boatswain?"
Q_1: "What pet did the poet Lord Byron had when he was a student at Trinity out of resentment for rules forbidding pet dogs like his beloved Boatswain?"
Q_2: "Who is the father of mathematician Ada Lovelace?"
E: "Lord Byron"
A: "A bear"
D_1: "Lord Byron also kept a tame bear while he was a student at Trinity out of resentment for rules forbidding pet dogs like his beloved Boatswain."

After imputation:
Q': "What pet did the poet and father of mathematician Ada Lovelace had when he was a student at Trinity?"
Q'_1 : "What pet did the poet Lord Byron had when he was a student at Trinity?"

Figure 6: Comparing Q pre- and post- imputation

the average perplexity score and in Figure 6 we showcase an example of how imputation leads to rephrasing sentences in a more natural way.

Table 4: Average perplexity of generated questions before and after imputation

Model	**PPL before imputation**	**PPL after imputation**
Synthetic grounded data	24.7	13.6
Synthetic ungrounded data	15.51	8.33

E.5 Non-monolithic setting: smaller models and and different models as data generators

We finetuned Llama3-8B instruct on 1250 synthetically-generated grounded examples resulting from the Source2Synth pipeline and 500 entries from HotpotQA. We used the 0-shot prompt from Figure 14 for evaluation and the soft-EM as a metric. Compared to the performance of the base model, LLMCurated shows an increase in accuracy of Ĩ2%.

Model	**Accuracy**
Base LLM (Llama3-8B)	57.8%
LLMSynth	64.46%
LLMCurated	71.13%

E.6 More results on prompt engineering

Table 5: **MHQA using different prompts.** Llama-2-70B-Chat accuracy across different prompts. *Role* "You are a QA-robot. Answer the following question:".

Prompt Type	**Model Accuracy (soft-EM, hotpotQA test set)**
0-shot	40.45%
Role	22.34%
1-shot	26.65%
Few-shots (5-shots)	21.83%
Role (1-shot)	28.29%

F SQL NON-EXECUTABLE CODE FILTERING

We discard incorrect SQL statements - i.e. whose execution with *sqlite3*[2] leads to an error. Discarded proportion: out of 50 tables, we generate 800 seed statements and the number of valid (executable) SQL statements was 658.

G PROMPTS

Zero-shot Table QA prompt.

Answer the following question using the table below. You may leverage an SQL tool.

{table}

Q: {question}

Figure 7: Zero-shot Table QA prompt for the TQA task.

One-Shot No context QA prompt.

– Example –
Q: What was the last year where this team was part of the US A-league?
A: 2004

Now do the same for the following question.
Q: {question}

One-shot Table QA prompt.

-- Example --
Answer the following question using the table below.
Your answer should be short and concise.

```
Season | Team         | League_apps | Goals
1923   |Swindon Town | 55          | 3
1922   |Swindon Town | 14          | 4
1921   |Swindon Town | 24          | 11
1920   |Swindon Town | 26          | 16
1919   |Swindon Town | 20          | 10
1914   |Swindon Town | 23          | 12
1913   |Swindon Town | 24          | 18
1912   |Swindon Town | 12          | 9
1911   |Swindon Town | 20          | 16
1910   |Swindon Town | 30          | 19
1909   |Swindon Town | 33          | 19
1908   |Swindon Town | 34          | 28
1907   |Swindon Town | 30          | 17
```

Q: How many league appearances were there between 1907 and 1909 (inclusive)?
A: 97

Now do the same for the following table and question.

{table}

Q: {question}

Figure 9: One-shot Table QA prompt for the TQA task.

[2]https://www.sqlite.org

One-shot Table+SQL QA prompt.

```
-- Example --
Answer the following question using the table below.
You may leverage an SQL tool.
The table is stored in a variable 'sql_table' and has the following schema:

Season | Team         | League_apps | Goals
1923   |Swindon Town | 55          | 3
1922   |Swindon Town | 14          | 4


Q: How many league appearances were there between 1907 and 1909 (inclusive)?


SQL: SELECT SUM(League_apps) FROM sql_table WHERE Season BETWEEN 1907 AND 1909

       | Result
result | 97



Now do the same for the following table and question.

{table}

Q: {question}
```

Figure 10: One-shot Table+SQL QA prompt for the TQA task.

Generating a seed in TQA.

```
Please generate an interesting statement about this table. The statement is a fact about one
of the columns in the following table.
{table}

An interesting statement as a result of this is:
```

Figure 11: Prompt used to induce a pertinent and interesting seed topic in TQA. This is done zero-shot.

Generating meaningful SQL in TQA.

```
Please generate SQL statements for the following table:

{table}

Seed: {seed}

An interesting SQL statement as a result of this is
```

Figure 12: Prompt used to induce a meaningful SQL statement given the table and seed for the TQA task. This is done zero-shot.

Generating a question in TQA.

```
I want to convert an SQL statement into a question.
Here is the original table:
{table}

SQL: {SQL}

What is the question that this SQL statement would be the answer to?
```

Figure 13: Prompt used to induce a meaningful question using the table and generated SQL query for the TQA task. This is done zero-shot.

Three-shot CoT prompt used at evaluation time on MHQA.

Answer the following multi-hop question ‘Q’ by decomposing it into ‘Q1’ and ‘Q2’ and solving them step-by-step. Learn from the following 3 examples. As shown in the following example:

-- Example #1 --
‘Q’ = ‘Who was the commander of the spaceflight that first landed humans on the Moon?’

1. Splitting ‘Q’ into ‘Q1’ and ‘Q2’:
‘Q1’ : ‘What was the spaceflight that first landed humans on the Moon?’;
‘Q2’ : ‘Who was the commander of [A1]?’;

2. Answering Q1:
The answer ‘A1’ to ‘Q1’ : ‘What was the spaceflight that first landed humans on the Moon?’ is ‘Apollo 11’. ‘A1’ = ‘Apollo 11’

3. Substituting A1 to Q2:
‘Q2’ : ‘Who was the commander of Apollo 11?’,

4. Answers Q2:
The answer ‘A2’ to Q2’ : ‘Who was the commander of Apollo 11?’ is ‘Neil Armstrong’.
‘A2’ = ‘A’ = ‘Neil Armstrong’

-- Example #2 --
‘Q’ = ‘What is the main ingredient in the flagship product of Ferrero?’

1. Splitting ‘Q’ into ‘Q1’ and ‘Q2’:
‘Q1’: ‘What is the flagship product of Ferrero?’
‘Q2’: ‘What is the main ingredient in [A1]?’

2. Answering Q1:
The answer ‘A1’ to ‘Q1’ : ‘What is the flagship product of Ferrero?’ is Nutella’.‘A1’ = Nutella’

3. Substituting A1 to Q2:
‘Q2’ : ‘What is the main ingredient in Nutella?’,

4. Answers Q2:
The answer ‘A2’ to Q2’ : ‘What is the main ingredient in Nutella?’.
‘A2’ = ‘A’ = ‘Hazelnuts

--Example #3 --

‘Q’ = ‘Who was the Roman Emperor when Jesus was born?’
1. Splitting ‘Q’ into ‘Q1’ and ‘Q2’:
‘Q1’: ‘When was Jesus born? ‘
‘Q2’: ‘Who was the Roman Emperor in [A1]?’

2. Answering Q1:
The answer ‘A1’ to ‘Q1’ : ‘When was Jesus born?’ is 1 BCE. ‘A1’ = 1 BCE

3. Substituting A1 to Q2:
‘Q2’ : ‘Who was the Roman Emperor in 1 BCE?’,

4. Answers Q2:
The answer ‘A2’ to Q2’ : ‘Who was the Roman Emperor in 1 BCE?’.
‘A2‘ = ‘A‘ = ‘Caesar Augustus‘

You MUST apply this structure when asked to answer a multi-hop question ‘Q’. Now answer the multi-hop question ‘Q‘ as shown in the examples above.
Q: {question}

Figure 14: *Three-shot CoT prompt* used at evaluation time in MHQA.

Prompt used to merge Q1 and Q2 in MHQA.

Merge ‘Q1‘ and ‘Q2’ into a single multi-hop bridge question ‘Q’.
Learn from the following 3 examples. As shown in the following example:

-- Example #1 --

‘Q1’ : "What was the spaceflight that first landed humans on the Moon?”
‘Q2’: "Who was the commander of Apollo 11?”

Solution:
1. Answer Q1; ‘A1’ is "Apollo 11”
2. If ‘A1’ is in ‘Q2’ print(A1); ‘A1’ = Apollo 11 is in ‘Q2’ so I print "Apollo 11”
3. Since you found ‘A1’ in ‘Q2’, rewrite ‘Q2’ so that you delete ‘A1’ and substitute ‘Q1’ there;
Rewriting Q2. Original ‘Q2’: "Who was the commander of Apollo 11?”. Since ‘A1’ is in ‘Q2’, I delete it and write ‘Q1’ there. Rewritten ‘Q2’: "Who was the commander of the spaceflight that first landed humans on the Moon?”

The single multi-hop question is therefore the rewritten ‘Q2’.
‘Q2‘ = ‘Q‘ = "Who was the commander of the spaceflight that first landed humans on the Moon?”

-- Example #2 --

‘Q1’: What is the flagship product of Ferrero?
‘Q2’: What is the main ingredient in Nutella?
Solution:
1. Answer Q1; ‘A1’ is "Nutella”
2. If ‘A1’ is in ‘Q2’ print(A1); ‘A1’ = "Nutella” is in ‘Q2’ so I print "Nutella”
3. Since you found ‘A1’ in ‘Q2’, rewrite ‘Q2’ so that you delete ‘A1’ and substitute ‘Q1’ there;
Rewriting Q2. Original ‘Q2’: "What is the main ingredient in Nutella?”.
Since ‘A1’ is in ‘Q2’, I delete it and write ‘Q1’ there.
Rewritten ‘Q2’: "What is the main ingredient in the flagship product of Ferrero?”

The single multi-hop question is therefore the rewritten ‘Q2’. ‘Q2’ = ‘Q’ = "What is the main ingredient in the flagship product of Ferrero?”

-- Example #3 --

‘Q1’: "When was Jesus born?”
‘Q2’: "Who was the Roman Emperor in 1 BCE?”

Solution:
1. Answer Q1; ‘A1’ is "1 BCE”
2. If ‘A1’ is in ‘Q2’ print(A1); ‘A1’ = 1 BCE is in ‘Q2’ so I print “1 BCE”
3. Since you found ‘A1’ in ‘Q2’, rewrite ‘Q2’ so that you delete ‘A1’ and substitute ‘Q1’ there;
Rewriting Q2. Original ‘Q2’: "Who was the Roman Emperor in 1 BCE?”. Since ‘A1’ is in ‘Q2’, I delete it and write ‘Q1’ there. Rewritten ‘Q2’: "Who was the Roman Emperor when Jesus was born?"

The single multi-hop question is therefore the rewritten ‘Q2’.
‘Q2’ = ‘Q’ = "Who was the Roman Emperor when Jesus was born?”

You MUST apply this structure when asked to merge ‘Q1’ and ‘Q2’.
Now merge ‘Q1’ and ‘Q2’ into a single multi-hop bridge question ‘Q’.
‘Q2’ : {question1}
‘Q2’ : {question2}

Figure 15: Prompt used to merge Q1 and Q2 in MHQA.

Generating Q1 in MHQA.

Identify one entity in the following text. Come up with a question so that the answer to this question is the entity chosen earlier. The question must be based on the following text. Write your results as 'Question:' and then the question and 'Entity:' and then the entity.

Text: {document_one}

Figure 16: Prompt used to generate Q_1. Q_1 is generated such that its answer $A1 = E$ where E is the entity retrieved.

Generating Q2 in MHQA.

Come up with a question based on the following text that contains the word: {entity}

Text: {document_two}

Figure 17: Prompt used to generate Q_2. Q_2 is generated such that its main topicis E where E is the entity retrieved.