Turning Tables: Generating Examples from Semi-structured Tables for Endowing Language Models with Reasoning Skills

Ori Yoran[†] **Alon Talmor** **Jonathan Berant**
Tel-Aviv University
{ori.yoran, alon.talmor, joberant}@cs.tau.ac.il

Abstract

Models pre-trained with a language modeling objective possess ample world knowledge and language skills, but are known to struggle in tasks that require reasoning. In this work, we propose to leverage semi-structured tables, and automatically generate at scale question-paragraph pairs, where answering the question requires reasoning over multiple facts in the paragraph. We add a pre-training step over this synthetic data, which includes examples that require 16 different reasoning skills such as number comparison, conjunction, and fact composition. To improve data efficiency, we sample examples from reasoning skills where the model currently errs. We evaluate our approach on three reasoning-focused reading comprehension datasets, and show that our model, PReasM, substantially outperforms T5, a popular pre-trained encoder-decoder model. Moreover, sampling examples based on model errors leads to faster training and higher performance.

Round	Date	Opponent	Venue	Result	Attendance
R3	31 October 1990	Portsmouth	H	0–0	16,699
R3R	6 November 1990	Portsmouth	A	3–2	16,085
R4	28 November 1990	Oxford United	A	2–1	9,789
QF	16 January 1991	Tottenham Hotspur	H	0–0	34,178
QFR	23 January 1991	Tottenham Hotspur	A	3–0	33,861
SF 1st Leg	24 February 1991	Sheffield Wednesday	H	0–2	34,074
SF 2nd Leg	27 February 1991	Sheffield Wednesday	A	1–3	34,669

Composition: q: *What was the Result when the Round was R4?* ***c*:** The Date when the Round was R4 was 28 November 1990. The Result when the Date was 28... **a: 2-1**

Comparison q: *Which Round had a higher Attendance: QF or QFR?* ***c:*** The Attendance when the Round was QF was 34,178. The Attendnace when the Round was QFR... **a: QF**

Date Difference: q: *The Opponent was Portsmouth how much time before the Opponent was Sheffield Wednesday?* ***c*:** The Date when the Opponent... **a: 3 months and 18 days**

Figure 1: An example table and question-context-answer triplets generated from the table as synthetic data. Each color corresponds to a different reasoning skill and colored cells are necessary to answer the question. The contexts shown are partial, such that the actual context contains the necessary information to answer the question and additional distractors. Answers are not necessarily extractive (e.g., date difference).

1 Introduction

Large pre-trained language models (LMs) (Devlin et al., 2019; Liu et al., 2019; Brown et al., 2020) have become the backbone of natural language processing in recent years. However, recent work has shown that they struggle in performing symbolic reasoning operations, such as composition or conjunction of facts (Talmor et al., 2019, 2020), numerical operations (Wallace et al., 2019; Hidey et al., 2020), and quantification (Warstadt et al., 2019), without substantial amounts of additional data.

Past work on improving reasoning in pre-trained models has taken two flavors: (a) adding specialized components for specific skills, like numerical and temporal reasoning (Ran et al., 2019; Gupta et al., 2020a; Khot et al., 2021; Chen et al., 2020a), or (b) generating synthetic examples at scale, for example, by using grammars or templates (Rozen et al., 2019; Zhao et al., 2019; Andreas, 2020; Asai and Hajishirzi, 2020; Campagna et al., 2020), and question generation models (Alberti et al., 2019; Puri et al., 2020; Bartolo et al., 2021).

In this work, we take the latter approach and argue that semi-structured tables are a valuable resource for automatic generation of training data that can endow LMs with reasoning skills. Tables can be crawled from the web at scale, and cover a wide range of domains and topics. Moreover, their structured nature makes them amenable to automatic processes of data generation. Specifically, given a table, we use templates to generate reading comprehension (RC) examples, that is, question-context-answer triplets, where answering the question requires diverse types of reasoning over facts mentioned in the context. Fig. 1 shows an example table, and three generated question-context-answer examples, which require fact composition, number comparison, and computing a date difference.

[†]Work done while working at the Allen Institute for Artificial Intelligence.

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics Volume 1: Long Papers, pages 6016 - 6031
May 22-27, 2022

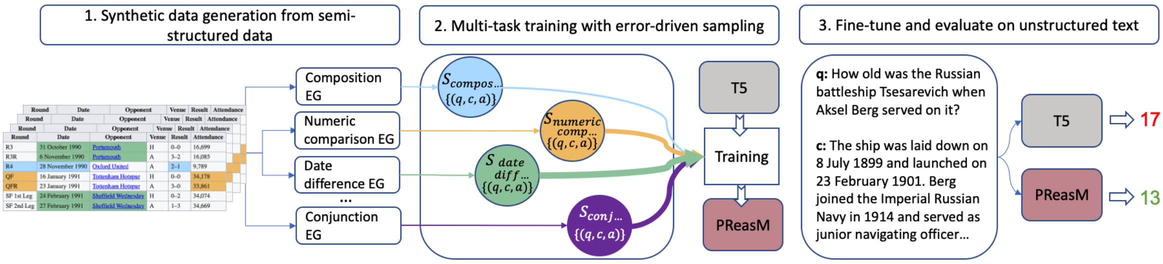

Figure 2: Approach overview. First, we use tables to generate large amounts of data from 16 different example generators (EGs), each corresponding to a different reasoning skill. Then, a pre-trained LM is trained over this data to obtain our model, PReasM, where we sample examples based on current model errors (arrow width corresponds to number of examples). Last, our model is fine-tuned and evaluated on target tasks that require reasoning.

Unlike prior work where semi-structured data was used for reasoning over tables or knowledge-bases (Eisenschlos et al., 2020; Yin et al., 2020; Herzig et al., 2020; Yu et al., 2021), here we harness tables to allow LMs to reason over *text* directly.

Fig. 2 provides an overview of our approach. We generate data by crawling tables from Wikipedia, and applying 16 different example generators (EGs) on each table. Each EG corresponds to a particular reasoning skill (composition, numerical comparison, see Table 1 for full list), and comprises a small set of question templates. Variables in the templates are filled with content from the table, and the structure of the table allows to compute the answer automatically. The context is a list of facts generated from the table that contain facts required for answering the question as well as distractor facts.

We add a pre-training step over this generated data, where we perform multi-task training over the 16 task corresponding to the EGs. Since each EG can generate vast numbers of examples, it is important to focus training on reasoning skills that the model lacks. Thus, we use *error-driven sampling* (Gottumukkala et al., 2020) to construct training batches, where most examples are sampled from EGs that the model currently struggles with.

We fine tune our **Pre**-traind for **Reas**oning **M**odel, PReasM, on three RC datasets that require reasoning: DROP (Dua et al., 2019), IIRC (Ferguson et al., 2020), and MMQA (Talmor et al., 2021). PReasM outperforms the original pre-trained T5 (Raffel et al., 2020) model by significant margins: 7.6, 4.1, and 1.2 F_1 points, respectively. Our results set a new state-of-the-art on MMQA and are the best results on IIRC for models where the retriever and reader are trained separately. Our analysis shows that PReasM leads to improvements of up to 40 F_1 points on specific question types, such as computing the difference between two dates, without causing a drop in other question types.

In conclusion, our results suggest that tables are a viable and untapped source of information for automatically generating large amounts of data that can be used to endow LMs with skills that are not captured using current pre-training approaches. Our code, data, and models are publicly available and can be downloaded from `https://github.com/oriyor/turning_tables`.

2 Data Generation

Our goal is to train a RC model that given a question q and textual context c returns an answer a, given a training set $D = \{(q_i, c_i, a_i)\}_{i=1}^{N}$. We focus on questions that require reasoning over the context, e.g., composing two facts. To endow LMs with reasoning skills, we want to generate a large synthetic training set $D_{syn} = \{(q_j, c_j, a_j)\}_{j=1}^{M}$ ($M \gg N$) from semi-structured tables, before fine-tuning on a target dataset.

2.1 Generating Examples from Tables

We use tables from English Wikipedia[1] to generate D_{syn}. English Wikipedia includes millions of tables with high lexical and domain diversity (Fetahu et al., 2019; Chen et al., 2020b; Gupta et al., 2020b; Talmor et al., 2021; Nan et al., 2021; Neeraja et al., 2021a). We extract from Wikipedia all tables $\mathcal{T}$ that have at least two columns and 10-25 rows, resulting in more than 700K tables. Then, we annotate all table columns with their semantic type (`STRING`, `NUMBER`, or `DATE`), which allows us to generate questions that involve manipulating numbers and dates. Details on the process of column annotation are in §A.1.

[1]We use the 01-01-2020 Wikipedia dump.

EG	*Template*	*Question*
2/3-hop Composition	What was the `col:1(s)` when the `col:2` was `val:2` in `table-title` of `page-title`?	"What was the **Play(s)** when the **Author** was **William Shakespeare** in **Notable works** of **Lidia Zamkow**?"
Conjunction	What was the `col:1` when the `col:2` was `val:2` and the `col:3` was `val:3` in `table-title` of `page-title`?	"What was the **Common name** when the **Family** was **Picidae** and the **Distribution** was **Okinawa** in **List of species** of **List of endemic birds of Japan**?"
Quantifiers Only	Is `val:1` the only `col:1` that has `col:2` `val:2` in `table-title` of `page-title`?	"Is **Jean Philippe** the only **Artist** that has **Language French** in **Results** of **Eurovision Song Contest 1959**?"
Quantifiers Every/Most	In `table-title` of `page-title`, does [OPERATOR] `col:1` have `col:2` `val:2`?	"In **Coal** of **List of Mines in South Africa**, does **every Mine** have **Owner Exxaro**?"
Num. Comparison	In `table-title` of `page-title`, which `col:1` had [OPERATOR] `col:2`: `val:1` or `val:1`?	"In **Administration** of **Mueang Nonthaburi District**, which **Name** had **a higher population**: **Suan Yai** or **Bang Khen**?"
Temp. Comparison	In `table-title` of `page-title`, what happened [OPERATOR]: the `col:1` was `val:1` or the `col:2` was `val:2`?	"In **Awards and nominations** of **Alexandre Pires**, what happened **earlier**: the **Category** was **Pop New Artist** or the **Category** was **Album of the Year**?"
Num. Yes/No Comparison	In `table-title` of `page-title` did `val:1` have [OPERATOR] `col:2` than `val:1`?	"In **Top employers** of **Chula Vista, California**, did **Walmart** have **more Employees** than **Target**?"
Temp. Yes/No Comparison	The `col:1` was `val:1` [OPERATOR] the `col:2` was `val:2` in `table-title` of `page-title`?	"The **Referee** was **Salim Oussassi more recently than when** the **Referee** was **Rachid Medjiba** in **1980 to 1999** of **Algerian Cup Final referees**?"
Temp./Num. Superlatives	In `table-title` of `page-title`, which `col:1` has the [OPERATOR] `col:2`?	"In **List of graphic novels** of **Minx (comics)**, which **Title** has the **earliest Release date**?"
Arithmetic Superlatives	In `table-title` of `page-title`, what was the [OPERATOR] `col:1` when the `col:2` was `val:2`?	"In **By rocket** of **1961 in spaceflight**, what was the **highest Successes** when the **Remarks** was **Maiden flight**?"
Counting	How many `col:1` have `col:2` `val:2` in `table-title` of `page-title`?	"How many **Elections** have **Candidate John Kufuor** in **Presidential elections** of **New Patriotic Party**?"
Arithmetic Addition	In `table-title` of `page-title`, what was the total number of `col:1` when the `col:2` was `val2`?	"In **Assists table** of **2010-11 La Liga**, what was the total number of **Assists** when the **Club** was **Villarreal**?"
Date Difference	In `table-title` of `page-title`, how much time had passed between when the `col:1` was `val:1` and when the `col:2` was `val:2`?	"In **Notable events | Concerts** of **Candlestick Park**, how much time had passed between when the **Artist** was **Paul McCartney** and when the **Artist** was **The Beatles**?"

Table 1: Question templates with examples for all EGs. Variable names specify permissible instantiations, where `col` is a column name, `val` is a value, and indices denote that a value must originate from a particular column. 2/3-hop composition examples are generated by generating 2/3-long fact chains between the answer and the value in the question. For example, above, the chain will include the facts *"The Role when the Author was Shakespeare was Lady Macbeth. The Play when the Role was Lady Macbeth was Macbeth"*. '[OPERATOR]' corresponds to EG-specific operators that we instantiate, e.g., in the EG 'Temp. comparison' [OPERATOR] is replaced with the operators *'earlier'* or *'later'*. Some EGs are collapsed into a single row (e.g., Quantifiers Every/Most).

The core of the generation process are the example generators (EGs), each corresponding to a reasoning skill (Table 1). Each example generator $g \in \mathcal{G}$ is a function that takes a table $t \in \mathcal{T}$ and randomly samples at most ten (q, c, a) triplets from the set of all possible triplets, where (i) q is a question is pseudo-language, (ii) c is the context, i.e., a list of facts extracted from t that includes the *gold facts* necessary for answering q and *distractor facts*, all phrased in pseudo-language, and (iii) a is the answer. Overall, the synthetic training set is $D_{syn} = \bigcup_{t \in \mathcal{T}} \bigcup_{g \in \mathcal{G}} g(t)$.

EGs generate examples in the following way. Each EG is associated with one or more question templates, which differ in their surface phrasing.[2] Templates contain typed variables that are instantiated with content from the table (see all variables in Table 1). Column and value variables are indexed to specify that the variable `val:i` must be instantiated by a value from the column `col:i`. Instantiating all variables results in the question q and the template allows us to programmatically compute the answer a. E.g., in the question from Fig. 1: *"In League Cup of 1990–91 Chelsea F.C. season, Which Round had a higher Attendance: QF or QFR?"* the answer a can be found by finding the rows with the values *"QF"* and *"QFR"* in the column *"Round"*, and returning the value that has a higher number in the column *"Attendance"*.

The context c is generated from the content necessary for answering the question, which can be identified using the instantiated question template. Facts generally have the form "The `col:1` when the `col:2` was `val:2` was `val:1`". E.g., to answer the question above, we generate the gold facts *"The Attendance when the Round was QF was 34,178"*, and *"The Attendance when the Round was QFR was 33,861"*. We also generate distractors by generating facts from rows or columns that are not relevant for the question, e.g., *"The Attendance when the Round was R4 was 9,789"*.

Overall, our process results in a large set D_{syn}, which includes examples from 16 EGs (all shown in Table 1).

[2] We also experimented with using just one question template per EG and observed very similar downstream results.

EG	*Question*	*Context*	*Answer*
3-hop Composition	What was the Result(s) when the Round was R4 in League Cup of 1990-91 Chelsea F.C. season?	In League Cup of 1990-91 Chelsea F.C. season: The attendance when the round was R2 1st Leg was 5,666. The result when the date was 6 November 1990 was 3-2. The date when the attendance was 34,669 was 27 February 1991. The attendance when the round was QF was 34,178. The date when the attendance was 34,074 was 24 February 1991. The date when the attendance was 16,085 was 6 November 1990. The attendance when the round was R3 was 16,699. **The date when the attendance was 9,789 was 28 November 1990. The result when the date was 28 November 1990 was 2-1.** The result when the date was 31 October 1990 was 0-0. The attendance when the round was QFR was 33,861. The result when the date was 16 January 1991 was 0-0. **The attendance when the round was R4 was 9,789.** The result when the date was 10 October 1990 was 4-1 (won 9-1 on agg). The date when the attendance was 5,666 was 26 September 1990.	2-1
Counting	In Presidential elections of New Patriotic Party, how many Elections have Candidate John Kufuor?	In Presidential elections of New Patriotic Party: The candidate when the election was 1992 was Albert Adu Boahen. The candidate when the election was 2008 (1) was Nana Akufo-Addo. **The candidate when the election was 2000 (2nd) was John Kufuor. The candidate when the election was 2000 (1st) was John Kufuor**. The candidate when the election was 1992 was Albert Adu Boahen. **The candidate when the election was 2004 was John Kufuor. The candidate when the election was 1996 was John Kufuor.** The candidate when the election was 2008 (2) was Nana Akufo-Addo.	4
Date Difference	In Notable concerts of Comiskey Park, how much time had passed between when the Artist was The Beatles and when the Artist was The Police?	In Notable concerts of Comiskey Park: The artist was Rush in August 19, 1979. **The artist was The Police in July 23, 1983.** The dates when the artist was The Jacksons were October 12, 1984, October 13, 1984, and October 14, 1984. The artist was Simon and Garfunkel in July 24, 1983. **The artist was The Beatles in August 20, 1965.** The date when the artist was Aerosmith was July 10, 1976.	17 years, 11 months, and 3 days

Table 2: Examples for generated (q, c, a) triplets. The first example is from the table in Fig. 1. Gold facts are bolded.

EG	*# Questions*	*EG*	*# Questions*
2-Hop composition	277,069	3-Hop composition	364,772
Conjunction	353,738	Only quantifier	522,071
Most quantifier	94,180	Every quantifier	16,693
Number comparison	410,749	Number Y/N comparison	410,749
Temporal comparison	453,499	Temporal Y/N comparison	470,694
Number superlatives	125,144	Temporal superlatives	80,884
Arithmetic superlatives	183,892	Arithmetic addition	86,969
Counting	484,471	Date difference	452,061
Total	4,787,635		

Table 3: Number of examples generated by each EG. During data generation, we randomly generate at most 10 examples from each EG and table.

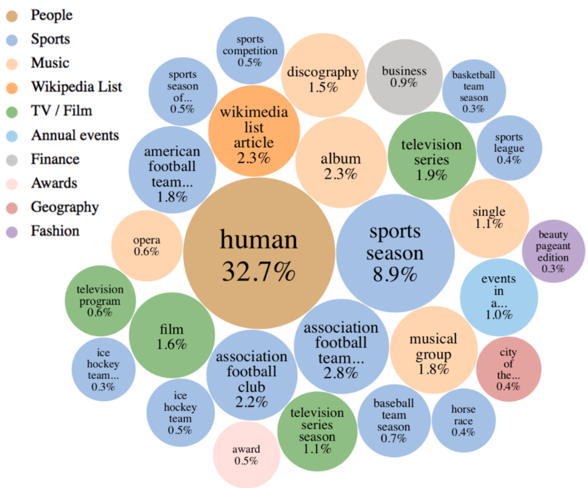

Figure 3: The most frequent categories of our Wikipedia pages and their frequency. Colors represent domains.

2.2 Data Analysis

Data generation yields 4.8M questions from over 176K tables and 130K pages. Table 2 contains examples for generated (q, c, a) triplets, including the full context c. Table 3 shows the number of generated examples for each EG. The number of distinct words is large (850K), illustrating the wide coverage and high lexical diversity of our approach. Moreover, generated examples have diverse answer types, which include text spans (43.2%), yes/no questions (31.6%), numeric (15.8%), and date answers (9.4%). In addition, our questions cover a wide range of domains including popular culture, politics and science. Tables cover more than 2,500 different Wikipedia categories, with 150 categories covering 80% of the data. Fig. 3 presents the most common categories of the Wikipedia pages from which we scraped our tables.

3 Training

Since our EGs generate large quantities of examples, one can think of each EG as providing an infinite stream of examples. In this setup, a natural question is how to construct training batches such that the model learns the required skills as quickly as possible. After briefly describing our model, we will detail our training framework, where we sample examples from EGs in an error-driven manner.

Model We use a standard encoder-decoder architecture (Raffel et al., 2020; Lewis et al., 2020). Given a training example (q, c, a), the model takes as input the sequence of tokens 'q [SEP] c', and the task is to autoregressively decode the answer a token-by-token. We train to maximize the maximum likelihood objective $\log P(a \mid q, c)$.

3.1 Multi-task Training over EGs

Given a pre-trained LM, we add another pre-training step, where we multi-task over a set of tasks $\mathcal{S}$, each task corresponding to examples generated from one EG. Similar to past work (Yogatama et al., 2019; Geva et al., 2020), to avoid "catastrophic forgetting" (Kirkpatrick et al., 2016) of the language skills, we sample batches from the original pre-training task with probability $\lambda = 0.5$.

Past work (Gottumukkala et al., 2020) has shown that *heterogeneous batching*, i.e., having examples from all tasks in each batch, leads to better performance compared to having entire batches from a single task. We follow this practice, and in every batch sample examples from every task according to a probability distribution $P_{tasks} \in \mathbb{R}^{|\mathcal{S}|}$. The main question is how to determine the distribution P_{tasks}, which we turn to next.

3.2 Sampling Strategies

We describe strategies for computing P_{tasks}, starting with the commonly-used *uniform sampling* approach, and then turn to error-driven approaches.

Uniform sampling Past work (Khashabi et al., 2020; Raffel et al., 2020; Wang et al., 2020) used uniform sampling, where the probability to sample from a task s is $P_{tasks}(s) = \frac{1}{|\mathcal{S}|}$, as a-priori all tasks are equally important. Some approaches also sample examples in proportion to the size of the training set (Raffel et al., 2020; Wang et al., 2020). This is not applicable in our case, where we assume an infinite stream of examples for every task, and make no assumptions on the distribution over reasoning skills in the downstream test set.

Error sampling Recent work (Sharma et al., 2018; Gottumukkala et al., 2020) proposed to construct P_{tasks} based on model errors, where one over-samples tasks with higher errors. More formally, let *Ceil*(s) be an estimate of the maximum accuracy achievable on a task s, and *Acc*(s) be the current model accuracy for task s on an held-out set. We define $\Delta(s) = \mathit{Ceil}(s) - \mathit{Acc}(s)$ and $P_{tasks}(s) = \frac{\Delta(s)}{\sum_{s' \in \mathcal{S}} \Delta(s')}$. The distribution P_{tasks} of a task is updated every time we evaluate the current model on the held-out data. In our setup, since the data is synthetic and abundant, we assume that the ceiling accuracy for all tasks is 1.0, and hence: $\Delta(s) = 1.0 - \mathit{Acc}(s)$. Similar to Gottumukkala et al. (2020), we use accuracy over a held-out set rather than the training loss, as this corresponds directly to our target metric.

Momentum sampling A potential issue with error sampling, is that if the error rate on a task is high, the model will spend most of its time on that task at the expense of other tasks, which may lead to low data efficiency. To remedy this, we introduce *momentum sampling*, a sampling strategy that samples from a task in proportion to its *rate of improvement*, putting most probability mass on skills that are improving rapidly.

Algorithm 1 Momentum Sampling(w, t, ϵ, k)

Input: windows size w, training time t, minimum share of examples per task ϵ, smoothing factor k.

for $s \in \mathcal{S}$ **do**
  if $t \geq w$ **then**
    $Acc_{\text{head}} \leftarrow \frac{1}{k}\sum_{i=t-k}^{t} Acc_s(i)$
    $Acc_{\text{tail}} \leftarrow \frac{1}{k}\sum_{i=t-w}^{t-w+k} Acc_s(i)$
    $P_{tasks}[s] \leftarrow \max(|Acc_{\text{head}} - Acc_{\text{tail}}|, \epsilon)$
  else
    $P_{tasks}[s] \leftarrow 1/|\mathcal{S}|$
$P_{tasks} \leftarrow P_{tasks}/\|P_{tasks}\|_1$

Alg. 1 provides the details of momentum sampling. Let t denote the index of a checkpoint evaluated on the held-out set, let w be a window size, and $Acc_s(i)$ be the held-out accuracy of checkpoint i on task s. We estimate model accuracy on a task s at the beginning and end of the window, and sample examples in proportion to the difference[3] in accuracy during that window. To smooth out accuracy fluctuations in adjacent checkpoints, we estimate accuracy as an average of k model checkpoints. During the first w checkpoint evaluations, we simply use uniform sampling.

Momentum sampling has several theoretical benefits over error sampling. First, it does not assume anything on the ceiling accuracy of a task. Second, when all tasks converge to their ceiling accuracy, momentum sampling converges to uniform sampling, unlike error sampling, which will over-sample from tasks for which *Ceil*(s) is low. This is useful in cases where the variance of *Ceil*(s) is high across tasks. On the other hand, momentum sampling requires a warm-up of w steps, and might under-sample from tasks that train slowly. In §A.2., we describe two synthetic experiments where momentum sampling clearly outperforms error sampling. However, we do *not* observe an advantage for momentum sampling in our experiments in §5, and leave further investigation of momentum sampling to future work.

4 Experimental Setup

4.1 Models

Baselines Our main baseline is T5 (Raffel et al., 2020), a popular pre-trained encoder-decoder model, which we fine-tune on the downstream

[3]We use the difference in performance and not the gain to account for cases of sudden drops in performance.

datasets. We experiment with *Base* and *Large* size models. For each dataset, we compare to the relevant state-of-the-art model.

Our pre-trained for reasoning model, PReasM, is a T5 model with another pre-training step on D_{syn}. We experiment with uniform sampling (*PReasM-Uni*), error sampling (*PReasM-Err*), and momentum sampling (*PReasM-Moment*) strategies.

4.2 Datasets

DROP (Dua et al., 2019) is a RC dataset with questions that require numeric reasoning. As an additional baseline, we also compare to GenBERT (Geva et al., 2020), which similar to our approach injects numerical skills by automatically generating synthetic data from a grammar.

IIRC (Ferguson et al., 2020) is a QA dataset, where annotators were given a single Wikipedia paragraph, and were asked to author questions that depend on that paragraph, but also on other paragraphs linked from the input paragraph. This resulted in questions that require discrete temporal (28%) or numeric (11%) reasoning. In addition, 30% of the questions are unanswerable.

We experiment with IIRC in two settings: (a) *Oracle*, where the model is given the gold context, reducing the problem to RC, where we can apply our models. (b) *Retrieval*, where we use the "improved pipeline"introduced by Ni et al. (2021) to retrieve the context, and replace the NumNet+ (Base) reader (Ran et al., 2019) used by the authors (which has specialized architecture) with T5/PReasM.

MMQA (Talmor et al., 2021) is a QA dataset, where the input is a question and a context that consists of a table, multiple text paragraphs, and multiple images, and the model must reason over a subset of the input modalities to answer the question.[4] We chose to use MMQA as it has many questions that involve a conjunction of facts, an operation that is largely missing from other datasets. Moreover, a large fractions of the questions can be answered by reasoning over the text and table only.

Since T5/PReasM cannot handle images or very long contexts, we construct a pipeline that automatically directs some MMQA questions to T5/PReasM, and uses the original *Implicit-Decomp* baseline from Talmor et al. (2021) elsewhere. Our pipeline includes two classifiers, the first determines whether a question requires reasoning over an image, and the second classifies whether a text paragraph is necessary to answer a question. Again, we experiment with an *oracle* and *retrieval* setting, such that in the oracle setting our model is presented with the gold paragraphs. We provide the full description of this pipeline in §A.4.

Evaluation metrics For all datasets, we use the official scripts for computing F_1 and EM, which compare the gold and predicted list of answers.

5 Experimental Results

We present results on the downstream RC datasets (§5.1) and on the synthetic data (§5.2).

5.1 Performance on RC Datasets

Table 4 presents the results of our large models over all datasets, also in comparison to current state-of-the-art. We observe that PReasM substantially improves performance compared to T5 in all conditions, improving on the test set by 7.6, 7.9, 4.1, and 1.2 F_1 points on DROP, $\text{IIRC}_{\text{oracle}}$, IIRC, and MMQA respectively.[5] We obtain new state-of-the-art results on MMQA and $\text{IIRC}_{\text{oracle}}$. On IIRC, we improve performance when using the same retriever (Pipeline) and replacing the NumNet+ reader with PReasM.[6] On DROP, specialized architectures for handling numbers still substantially outperform both T5 and PReasM.

Table 5 shows the effect of different sampling strategies. Error sampling and momentum sampling generally outperform uniform sampling, but there is no clear advantage to momentum sampling compared to error sampling. We further analyze the effect of sampling methods in §5.2.

We now look at performance on different answer types across datasets, where PReasM leads to dramatic improvements on some types, while maintaining similar performance on other types.

DROP Table 6 breaks down performance based on answer types: PReasM outperforms T5 across

[4]We removed tables that appear in the MMQA development and test sets from D_{syn}.

[5]To verify that the gains of PReasM over T5 are not due to knowledge memorized from D_{syn}, we trained T5 and PReasM to generate the answer given the question only (without context). We found that the performance of T5 and PReasM is nearly identical in this setup.

[6]We report the numbers from Ni et al. (2021) (45.8/44.3 F_1 on the development/test sets). To fairly compare with the NumNet+ reader, we got the retrieved paragraphs for the Pipeline model through personal communication. However, results on these paragraphs was lower than reported in the paper: 45.5/42.8 F_1. The reported results of our models are with this slightly worse retriever, but still outperform the performance of NumNet+ (Pipeline) from the original paper.

Dataset	Model	Development	Test
DROP	T5-Large	64.6/61.8	65.0/61.8
	PReasM-Large	**72.3/69.4**	**72.6/69.5**
	GenBERT	72.3/68.8	72.4/68.6
	QDGAT-ALBERT		**90.1/87.0**
$\text{IIRC}_{\text{oracle}}$	T5-Large	69.9/64.9	67.1/62.7
	PReasM-Large	**77.4/72.7**	**75.0/70.6**
	NumNet+	69.2/63.9	70.3/65.6
IIRC	T5-Large (Pipeline)	47.4/44.2	41.0/37.8
	PReasM-Large (Pipeline)	**50.0/46.5**	**45.1/42.0**
	NumNet+ (Pipeline)	45.8/41.7	44.3/41.3
	NumNet+ (Joint)	**50.6/46.9**	**50.5/47.4**
MMQA	T5-Large	64.3/57.9	63.4/57.0
	PReasM-Large	**65.5/59.0**	**64.6/58.3**
	Implicit-Decomp	55.5/48.8	55.9/49.3

Table 4: Development and test results. The two values in each cell indicate F_1/EM. Improvement over T5 is statistically significant in all cases ($p < 0.05$) according to the paired bootstrap test (Efron and Tibshirani, 1993).

Model	DROP	$\text{IIRC}_{\text{oracle}}$	IIRC	MMQA
T5-Large	64.6±0.1	69.6±0.3	46.7±0.5	64.2±0.2
PReasM-Uni-Large	71.4±0.1	75.1±0.2	48.9±0.3	64.9±0.4
PReasM-Moment-Large	71.7±0.1	**76.8**±0.4	**49.7**±0.1	64.9±0.2
PReasM-Err-Large	**72.2**±0.1	76.5±0.5	49.3±0.4	**65.3**±0.1

Table 5: F_1 on the development set with different sampling strategies. Results show the average and standard deviation over 3 seeds for DROP and MMQA, and 5 seeds for IIRC and $\text{IIRC}_{\text{oracle}}$.

the board for all model sizes and answer types. PReasM-Base outperforms GenBERT on 3 of 4 answer types. The high performance of GenBERT on *Number* questions can be explained by: (a) GenBERT uses digit tokenization which improves arithmetic reasoning (Thawani et al., 2021), and (b) training on multiple numerical reasoning templates.

IIRC Table 7 breaks down performance based on answer types. PReasM outperforms T5 in the oracle setup by roughly 8 points for both Base and Large models, and by 2.6-4 points in the retrieval setup. Improvements are mostly due to cases when the answer is a numerical *Value*, where PReasM outperforms T5 by 39.1 and 40.3 F_1 points in Base and Large models (oracle setup).

Comparing PReasM-Base to NumNet+, PReasM outperforms NumNet+ on *None*, *Span* and *Binary* questions, but lags behind on *Value* questions, where NumNet+ uses specialized architecture.

Overall, PReasM-Large improves state-of-the-art in the oracle setup by 4.7 F_1 points. In the retrieval setting, PReasM outperforms NumNet+ (Pipeline) by 4.2 and 0.8 F_1 points on the development and test sets, respectively (see Table 4).

Model	Span	Spans	Date	Number	Total
T5-Base	77.5	65.8	57.1	43.7	55.8
PReasM-Base	**81.1**	**69.4**	**64.6**	**61.5**	**68.1**
T5-Large	86.1	**78.4**	75.7	52.2	64.6
PReasM-Large	**86.6**	**78.4**	**77.7**	**64.4**	**72.3**
GenBERT	74.5	24.2	56.4	**75.2**	**72.3**

Table 6: Drop development F_1 across answer types.

Model	Oracle	None	Span	Binary	Value	Total
T5-Base	✓	91.4	72.0	76.6	8.7	66.3
PReasM-Base	✓	92.5	74.9	71.9	47.8	**74.5**
T5-Large	✓	92.2	77.7	81.3	10.9	69.9
PReasM-Large	✓	92.2	78.4	80.5	51.2	**77.4**
T5-Base	✗	57.1	47.6	54.7	6.7	43.5
PReasM-Base	✗	53.9	49.1	64.8	24.3	**47.5**
T5-Large	✗	56.2	49.9	77.3	11.5	47.4
PReasM-Large	✗	55.9	50.8	69.5	28.6	**50.0**
NumNet+ (Pipeline)	✗	49.6	48.4	52.3	30.0	45.8

Table 7: IIRC Development F_1 across answer types.

MMQA Table 8 breaks down performance based on reasoning skills (annotated per example in MMQA). PReasM outperforms T5 in both the oracle and retrieval setting, and for both model sizes.

The main cause for improvement are comparison questions, where PReasM outperforms T5 by 19 and 11.7 F_1 points on Base and Large models. PReasM outperforms T5 on conjunction questions in Base models, and yes/no questions in all settings. Interestingly, T5 is equipped with decent composition skills, *without* any specialized pre-training.

Compared to *Implicit-Decomp*, although *Implicit-Decomp* outperforms our models on questions that require hopping between two table columns and aggregations, PReasM outperforms *Implicit-Decomp* in all other cases. When considering only questions that require reasoning over text and tables, PReasM-Large improves F_1 by 16.1 points, from 62.3 to 78.4.

5.2 Performance on D_{syn}

Fig. 4 shows statistics on the performance of PReasM on different tasks in D_{syn} during training. The average accuracy across all tasks at the end of training is high – almost 98.0 F_1. PReasM reaches high performance on all tasks, where the lowest-performing tasks are 'arithmetic addition' (91.1) and 'date difference' (94.7). On those tasks, the advantage of error-driven sampling is evident, and it outperforms uniform sampling by as much as 4 points.

Zooming-in on the learning curve, momentum

Model	*Oracle*	*ColumnHop*	*Text*	*Composition*	*Comparison*	*Conjunction*	*Yes/No*	*Aggregate*	*Total*
T5-Base	✗	81.7	75.2	67.0	61.8	74.1	76.9	27.3	71.9
PReasM-Base	✗	80.8	75.7	66.3	80.8	80.8	83.1	36.4	74.3
T5-Large	✗	82.6	79.8	71.8	69.3	83.0	83.1	27.3	76.8
PReasM-Large	✗	84.0	79.7	71.9	81.0	82.3	93.8	36.4	78.4
T5-Base	✓	85.2	82.1	74.6	63.3	77.4	80.0	27.3	77.9
PReasM-Base	✓	86.9	80.0	75.4	84.1	82.6	89.2	36.4	79.9
T5-Large	✓	88.2	85.9	79.4	74.1	83.2	83.1	36.4	82.7
PReasM-Large	✓	87.8	85.6	79.8	83.6	82.3	90.8	45.5	83.8
Implicit-Decomp	✓	96.6	57.1	53.2	78.4	68.1	76.9	59.1	62.3

Table 8: Development F_1 on MMQA with reasoning type breakdown on the development set. The column 'Total' refers to all questions that do not require reasoning over the image modality.

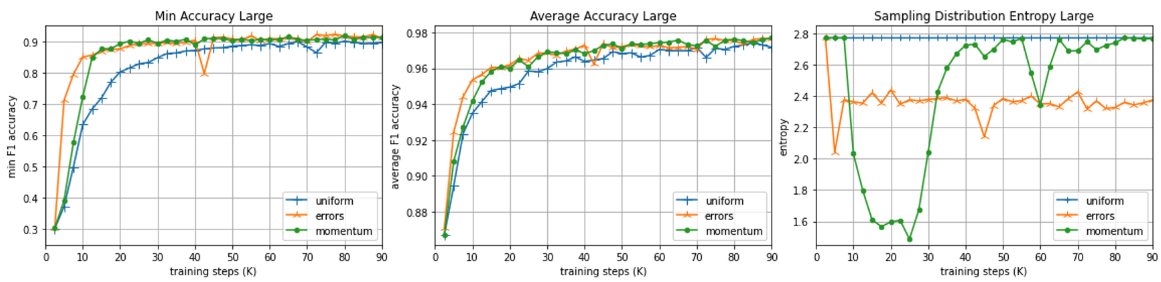

Figure 4: Minimum (left) and average (center) task accuracy on 1,000 held-out examples per task from D_{syn}, and the entropy of P_{tasks} (right) as a function of the number of training steps for all sampling strategies (Large models).

and error sampling learn reasoning skills a lot faster than uniform sampling. Looking at the entropy of P_{tasks} sheds light on the difference between error sampling and momentum sampling. Error sampling puts most probability mass on the lowest-performing task (arithmetic addition), and thus its entropy over tasks is roughly constant from a certain point. Conversely, momentum sampling puts a lot of probability mass on tasks that are improving quickly at the beginning, but as improvements plateau, it converges towards uniform sampling.

Fig. 5 and Table 11 (in the Appendix) show the results for T5 and PReasM on D_{syn}. The results for T5 were obtained by training in a few-shot manner on 32 examples for 200 steps, as suggested in Ram et al. (2021). T5-Large outperforms T5-Base on most tasks, suggesting that larger models are able to learn reasoning skills faster. On tasks such as date difference and arithmetic addition, the results for T5-Large are low, at around 10 F_1. Our PReasM models significantly outperform T5 on all tasks.

6 Analysis

Reasoning skills in DROP To check which reasoning skills PReasM has, we use a proposed split of a subset of DROP to reasoning skills (Gupta et al., 2020a). Table 9 presents the F_1 for our best PReasM and T5 models, as well as the F_1 from

Question Type	*NMN*	T5-*Base*	PReasM-*Base*	T5-*Large*	PReasM-*Large*
Date-Compare	82.6	86.4	87.5	87.6	**89.9**
Date-Difference	75.4	19.6	78.9	45.4	**80.4**
Number-Compare	92.7	91.3	95.2	97.3	**98.5**
Extract-Number	86.1	91.8	94.9	92.1	**95.1**
Count	55.7	80.1	86.7	86.7	**89.2**
Extract-Argument	69.7	87.6	86.2	90.5	**92.1**

Table 9: F_1 on a previously-proposed split of a subset of the development set of DROP to reasoning skills.

the neural module network (NMN) used in Gupta et al. (2020a). NMN was trained only on a subset of the original DROP dataset. When comparing to T5, PReasM dramatically improves performance on Date-Difference, and also leads to sizable gains in Number-Compare, Extract-Number and Count.

Accuracy vs. training cost trade-off We evaluate PReasM-Base models on DROP and $IIRC_{oracle}$ as we vary the number of pre-training steps on D_{syn} (Fig. 6). Most of the improvement happens in the first 100K steps, and error-driven sampling outperforms uniform sampling throughout training. Error sampling outperforms momentum sampling in the latter part of training. A possible reason is that the reasoning skills in the downstream tasks are correlated with the harder tasks during pre-training (arithmetic addition and date difference). This provides an advantage for error

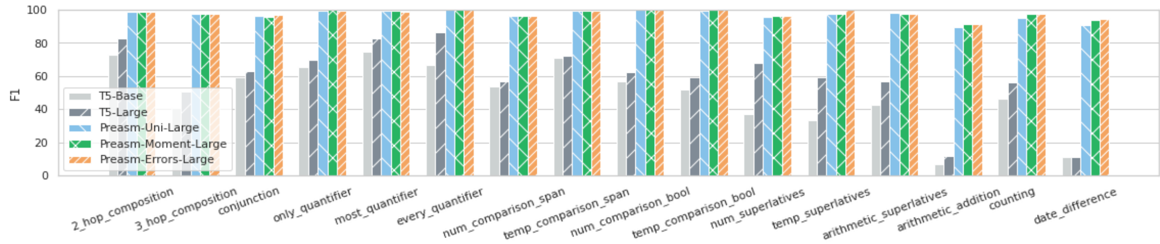

Figure 5: F_1 for each task in D_{syn}, for T5 and PReasM on the held-out evaluation set.

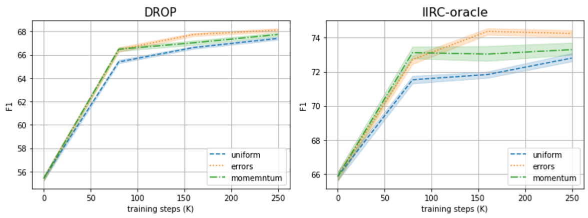

Figure 6: Development F_1 on DROP and $\text{IIRC}_{\text{oracle}}$ as a function of the number of training steps (Base models). The light lines mark confidence intervals over 5 seeds. The first point shows the performance of T5-Base.

sampling, since it will focus on these tasks even if the improvement during pre-training is small.

7 Related Work

Template-based data generation has been previously used for data augmentation, for example to inject numerical skills (Geva et al., 2020), and to improve consistency (Asai and Hajishirzi, 2020), and zero-shot accuracy (Zhao et al., 2019). In addition, templates were used for dataset construction (Talmor and Berant, 2018; Clark et al., 2020; Thorne et al., 2021), and to analyse model generalization (Rozen et al., 2019). In this work, we automatically generate examples by instantiating templates using structured data. Since our method relies solely on tables as input, it is highly scalable, has rich lexical diversity, and can be easily extended to new skills and domains.

Recently, Thorne et al. (2021) introduced the WIKINLDB dataset, which includes queries that require reasoning over a set of textual facts. Queries are instantiated with values from a knowledge graph (KG), and facts are generated by a LM. Unlike this work, WIKINLDB is focused on *evaluating* reasoning skills. We, on the other hand, show that generated examples can be used to endow a pre-trained LM with new reasoning skills. Moreover, tables are much easier to collect at scale compared to KGs, which tend to have limited coverage.

Data augmentation techniques have been extensively explored in RC, QA, and dialogue (Feng et al., 2021; Talmor and Berant, 2019; Khashabi et al., 2020; Alberti et al., 2019; Puri et al., 2020; Bartolo et al., 2021). Here, we focus on tables as a valuable source for data generation.

Pre-training over tables has focused in the past on reasoning over tables and knowledge-bases (Eisenschlos et al., 2020; Yin et al., 2020; Herzig et al., 2020; Müller et al., 2021; Yu et al., 2021; Neeraja et al., 2021b). Here, we use pre-training over tables to improve reasoning over *text*. We leave evaluation on tasks beyond RC to future work.

Error-driven sampling has been considered in the past in the context of active learning (Sharma et al., 2018), reinforcement learning (Graves et al., 2017; Glover and Hokamp, 2019; Xu et al., 2019), transfer learning (Zhang et al., 2020; Pilault et al., 2021), and distributionally robust optimization (Oren et al., 2019; Sagawa et al., 2020), where the goal is to perform well over a family of distributions. Similar to Gottumukkala et al. (2020), we compute heterogeneous batches based on error rates, and show that this improves efficiency and performance.

8 Conclusion

We propose semi-structured tables as a valuable resource for generating examples that can endow pre-trained language models with reasoning skills. We generate 5M examples that correspond to 16 reasoning skills from Wikipedia tables and add a pre-training step over this data. To improve data efficiency we use error-driven sampling, which focuses training on reasoning skills that the model currently lacks. We evaluate our model, PReasM, on three reasoning-focused RC datasets and show that it leads to substantial improvements in all cases.

Acknowledgments

We thank Elad Segal, Uri Shaham, Tomer Wolfson, and Ankit Gupta for their useful comments and James Ferguson, Ansong Ni and Matt Gardner for their help with the IIRC dataset. This research was partially supported by The Yandex Initiative for Machine Learning, and the European Research Council (ERC) under the European Union Horizons 2020 research and innovation programme (grant ERC DELPHI 802800).

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Akari Asai and Hannaneh Hajishirzi. 2020. Logic-guided data augmentation and regularization for consistent question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5642–5650, Online. Association for Computational Linguistics.

Max Bartolo, Tristan Thrush, Robin Jia, Sebastian Riedel, Pontus Stenetorp, and Douwe Kiela. 2021. Improving question answering model robustness with synthetic adversarial data generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8830–8848, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica Lam. 2020. Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 122–132, Online. Association for Computational Linguistics.

Kunlong Chen, Weidi Xu, Xingyi Cheng, Zou Xiaochuan, Yuyu Zhang, Le Song, Taifeng Wang, Yuan Qi, and Wei Chu. 2020a. Question directed graph attention network for numerical reasoning over text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6759–6768, Online. Association for Computational Linguistics.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020b. Tabfact: A large-scale dataset for table-based fact verification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3882–3890. ijcai.org.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Number 57 in Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, Boca Raton, Florida, USA.

Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online. Association for Computational Linguistics.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, Online. Association for Computational Linguistics.

James Ferguson, Matt Gardner, Hannaneh Hajishirzi, Tushar Khot, and Pradeep Dasigi. 2020. IIRC: A dataset of incomplete information reading comprehension questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1137–1147, Online. Association for Computational Linguistics.

Besnik Fetahu, Avishek Anand, and Maria Koutraki. 2019. Tablenet: An approach for determining fine-grained relations for wikipedia tables. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2736–2742. ACM.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, Online. Association for Computational Linguistics.

John Glover and Chris Hokamp. 2019. Task selection policies for multitask learning.

Ananth Gottumukkala, Dheeru Dua, Sameer Singh, and Matt Gardner. 2020. Dynamic sampling strategies for multi-task reading comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 920–924, Online. Association for Computational Linguistics.

Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1311–1320. PMLR.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2020a. Neural module networks for reasoning over text. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020b. INFOTABS: Inference on tables as semi-structured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324, Online. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Christopher Hidey, Tuhin Chakrabarty, Tariq Alhindi, Siddharth Varia, Kriste Krstovski, Mona Diab, and Smaranda Muresan. 2020. DeSePtion: Dual sequence prediction and adversarial examples for improved fact-checking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8593–8606, Online. Association for Computational Linguistics.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Tushar Khot, Daniel Khashabi, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2021. Text modular networks: Learning to decompose tasks in the language of existing models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1264–1279, Online. Association for Computational Linguistics.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. *ArXiv preprint*, abs/1612.00796.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Thomas Müller, Julian Eisenschlos, and Syrine Krichene. 2021. TAPAS at SemEval-2021 task 9: Reasoning over tables with intermediate pre-training. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 423–430, Online. Association for Computational Linguistics.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryscinski, Nick Schoelkopf, Riley Kong, Xiangru Tang, Murori Mutuma, Ben Rosand, Isabel Trindade, Renusree Bandaru, Jacob Cunningham, Caiming Xiong, and Dragomir R. Radev. 2021. Fetaqa: Free-form table question answering. *ArXiv preprint*, abs/2104.00369.

J. Neeraja, Vivek Gupta, and Vivek Srikumar. 2021a. Incorporating external knowledge to enhance tabular reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2799–2809, Online. Association for Computational Linguistics.

J. Neeraja, Vivek Gupta, and Vivek Srikumar. 2021b. Incorporating external knowledge to enhance tabular reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2799–2809, Online. Association for Computational Linguistics.

Ansong Ni, Matt Gardner, and Pradeep Dasigi. 2021. Mitigating false-negative contexts in multi-document question answering with retrieval marginalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6149–6161, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yonatan Oren, Shiori Sagawa, Tatsunori B. Hashimoto, and Percy Liang. 2019. Distributionally robust language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4227–4237, Hong Kong, China. Association for Computational Linguistics.

Jonathan Pilault, Amine Elhattami, and Christopher J. Pal. 2021. Conditionally adaptive multi-task learning: Improving transfer learning in NLP using fewer parameters & less data. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5811–5826, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Ori Ram, Yuval Kirstain, Jonathan Berant, Amir Globerson, and Omer Levy. 2021. Few-shot question answering by pretraining span selection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3066–3079, Online. Association for Computational Linguistics.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. NumNet: Machine reading comprehension with numerical reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2474–2484, Hong Kong, China. Association for Computational Linguistics.

Ohad Rozen, Vered Shwartz, Roee Aharoni, and Ido Dagan. 2019. Diversify your datasets: Analyzing generalization via controlled variance in adversarial datasets. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 196–205, Hong Kong, China. Association for Computational Linguistics.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. 2020. Distributionally robust neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Sahil Sharma, Ashutosh Kumar Jha, Parikshit Hegde, and Balaraman Ravindran. 2018. Learning to multitask by active sampling. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4911–4921, Florence, Italy. Association for Computational Linguistics.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. oLMpics-on what language model pre-training captures. *Transactions of the Association for Computational Linguistics*, 8:743–758.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Alon Talmor, Ori Yoran, Amnon Catav, Dan Lahav, Yizhong Wang, Akari Asai, Gabriel Ilharco, Hannaneh Hajishirzi, and Jonathan Berant. 2021. Multimodalqa: complex question answering over text, tables and images. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Avijit Thawani, Jay Pujara, Filip Ilievski, and Pedro Szekely. 2021. Representing numbers in NLP: a survey and a vision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–656, Online. Association for Computational Linguistics.

James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Halevy. 2021. Database reasoning over text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3091–3104, Online. Association for Computational Linguistics.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315, Hong Kong, China. Association for Computational Linguistics.

Xinyi Wang, Yulia Tsvetkov, and Graham Neubig. 2020. Balancing training for multilingual neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8526–8537, Online. Association for Computational Linguistics.

Alex Warstadt, Yu Cao, Ioana Grosu, Wei Peng, Hagen Blix, Yining Nie, Anna Alsop, Shikha Bordia, Haokun Liu, Alicia Parrish, Sheng-Fu Wang, Jason Phang, Anhad Mohananey, Phu Mon Htut, Paloma Jeretic, and Samuel R. Bowman. 2019. Investigating BERT's knowledge of language: Five analysis methods with NPIs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2877–2887, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Yichong Xu, Xiaodong Liu, Yelong Shen, Jingjing Liu, and Jianfeng Gao. 2019. Multi-task learning with sample re-weighting for machine reading comprehension. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2644–2655, Minneapolis, Minnesota. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.

Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. Learning and evaluating general linguistic intelligence.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. 2021. Grappa: Grammar-augmented pre-training for table semantic parsing. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Sheng Zhang, Xin Zhang, Weiming Zhang, and Anders Søgaard. 2020. Worst-case-aware curriculum learning for zero and few shot transfer.

Zijian Zhao, Su Zhu, and Kai Yu. 2019. Data augmentation with atomic templates for spoken language understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3637–3643, Hong Kong, China. Association for Computational Linguistics.

A Supplemental Material

A.1 Data Generation

In this section, we provide details about how we classify table columns.

Classifying table columns When annotating the semantic types of columns, a column will be of type `NUMBER` or `DATE` if all values in the column can be parsed with standard tools for parsing numbers and dates,[7] accordingly. Otherwise, we annotate the column as type `STRING`.

Information in tables is usually aggregated such that certain columns serve as the *semantic index* of the table. For example, the table in Fig. 1 provides information about each round in a tournament. In order for our examples to be semantically meaningful, we generate questions about columns that serve as the semantic index of their table.

[7] `https://pypi.org/project/python-dateutil/`

Since the semantic index is not provided, we use a linear decision rule to find such columns. The features to our classifier include the column's distance from the leftmost column, the percentage of unique cells in the column, the percentage of cells whose values are links to Wikipedia articles, the percentage of cells with short text (at most 2 characters), the percentage of cells with numbers, and the column's header. We allow more than one semantic index per table, such that both the *Round* and *Opponent* columns can serve as a semantic index in the table in Fig. 1.

A.2 Advantages of Momentum Sampling

To highlight the theoretical benefits of momentum sampling, we construct synthetic experiments where there is high variance in the ceiling accuracy between different tasks. As we show in §5.2, our models are able to achieve near perfect accuracy on our tasks when provided with enough training examples. Hence, we create settings where the ceiling accuracy for a task is lower than 1.0, either by adding noise or by down-sampling the number of training examples. More specifically, we train on two tasks: an *arithmetic addition* task that trains slowly and has a high ceiling accuracy, and a second task that trains quickly, and evaluate the performance on a held-out set of arithmetic addition examples.

First, we train on *arithmetic addition* and *2-hop composition*, which is faster to train. We conduct two experiments, in which we add noise to the 2-hop composition task by randomly sampling the label from the vocabulary in order to force the ceiling accuracy to be lower than 1.0. To check the performance of sampling strategies in varying levels of noise, we conduct two experiments where we add noise to 30% or 100% of the examples (in the latter case the label of 2-hop composition is random). We expect that this will lead to slower learning of arithmetic addition for error sampling, since more probability mass will be allocated to the noisy task (since its ceiling accuracy is low), despite the fact that it is easier.

Next, we train on *arithmetic addition* and *date difference*, both of which train slowly. To force the ceiling accuracy of the date difference task to be lower than 1.0, our training set contains only 1,000 examples. This emulates settings where the data is not generated automatically and the cost of generating examples is higher. Again, we expect error sampling to over-sample from the date difference task even when this would not lead to gains in performance, due to the small training set.

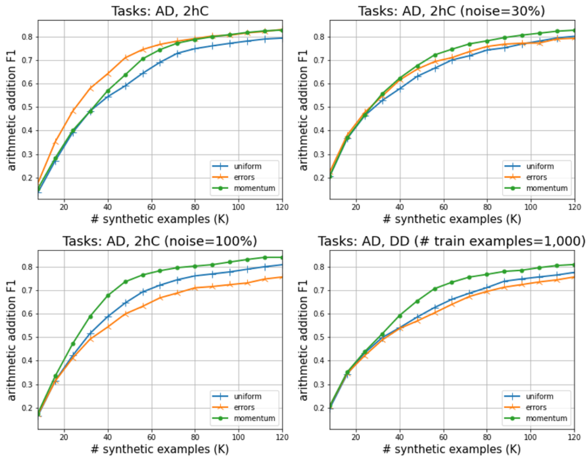

Figure 7: Motivation for momentum sampling. AD=Arithmetic Addition, 2hC=2-hop Composition, DD=Date Difference. When one task has high ceiling accuracy and trains slowly, and the other task has a lower ceiling accuracy and trains fast, momentum sampling outperforms error sampling.

Fig. 7 illustrates the advantage of momentum sampling in these settings. Without noise (top left), both momentum sampling and error sampling learn faster than uniform sampling. Momentum sampling learns more slowly than error sampling, due to the warm-start period in the first w evaluation checkpoints. As we add noise to 30% of the examples (top right), error sampling focuses on the noisy task once accuracy approaches a certain level (0.7 F_1). When we add noise to all of the 2-hop composition examples (bottom left), uniform sampling outperforms error sampling, while momentum sampling still performs well. This phenomenon repeats when we switch the 2-hop composition task with the date difference task and down-sample the number of training examples (bottom right).

A.3 Implementation Details

The following section includes implementation details for our experiments, including: hyperparameters for the momentum sampling algorithm, the original pre-training task, and technical details.

Experiment	*Size*	*LR*	*Batch Size*	*GAS*	*Epochs*
PReasM	Base	1e-4	64	1	50
PReasM	Large	1e-4	18	4	36
DROP	Base	1e-4	20	1	20
IIRC	Base	1e-4	20	1	60
$\text{IIRC}_{\text{oracle}}$	Base	1e-4	20	1	60
MMQA	Base	1e-4	6	3	20
DROP	Large	5e-5	16	2	20
IIRC	Large	5e-5	16	2	60
$\text{IIRC}_{\text{oracle}}$	Large	5e-5	16	2	60
MMQA	Large	1e-4	2	16	10

Table 10: Hyper-parameters used in all experiments, LR and GAS refer to learning-rate and gradient accumulation steps. In our PReasM experiments, epochs refer to the number of steps between evaluations, which is set to $5,000$ and $2,500$ for our base and large experiments, which leads to $250,000$ and $90,000$ optimization steps, respectively.

Momentum sampling For momentum sampling we use a window size of $w = 4$, a smoothing factor of $k = 2$, and sample at least $\epsilon = 0.002$ examples from every task in D_{syn}.

Original pre-training task In order to avoid catastrophic forgetting (Kirkpatrick et al., 2016), we continue training with the span-corruption objective introduced in (Raffel et al., 2020), over sequences of length 256 from the English Wikipedia.

Technical details We train all our experiments on one RTX8000 (48GB) or RTX3090 (24GB) GPUs. Our PReasM-Base and PReasM-Large models training time was 5-6 and 8-9 days on one RTX8000 GPU, respectively. We use the T5 model from `https://huggingface.co/transformers/model_doc/t5.html` (Wolf et al., 2020). Table 10 contains the hyper-parameters used in our experiments.

A.4 MMQA Pipeline

The first classifier in our pipeline is a T5-large model fine-tuned on the MMQA training set to determine if a question is likely to require an image or not. When the classifier determines a question requires an image, the example is directed to *Implicit-Decomp*. The accuracy of this classifier on the MMQA development set is 99.2%.

The second classifier in the pipeline is a T5-3B model, fine-tuned on the MMQA training set to determine given a question and one of the textual paragraphs if that paragraph is required for answering the question. Then, for every question that does not require an image, we classify each of the textual paragraphs and only use the ones classified as relevant. This process identifies all gold paragraphs in 95.8% of the examples.

Last, we convert the table into text by linearizing the table as described in Talmor et al. (2021). The model is presented with multiple paragraphs and the linearized table, and can answer questions that require any reasoning across them. Since the context is long, we present the model with contexts of size 1,536 word-pieces (without any change to the original T5 model).

	T5-Base	*PReasM-Uni-*Base	*PReasM-Moment-*Base	*PReasM-Err-*Base	T5-Large	*PReasM-Uni-*Large	*PReasM-Moment-*Large	*PReasM-Err-*Large
2-hop Composition	72.8	98.6	98.4	98.6	82.6	98.5	98.5	98.5
3-hop Composition	40.7	97.5	97.9	97.3	50.8	97.6	97.5	97.6
Conjunction	59.6	96.1	95.9	95.9	63.2	96.5	96	96.7
Quantifiers Only	65.8	99.8	99.9	99.5	69.6	99.7	100	99.7
Quantifiers Most	74.7	99.6	99.2	99	82.5	99.6	99.7	99.4
Quantifiers Every	67	100	100	100	86.6	100	100	100
Numerical Comparison	53.6	96.3	96.6	96.6	57.1	96.6	96.5	96.5
Temporal Comparison	71.1	99.3	99.2	99.2	72.3	99.3	99.2	99.4
Numerical Comparison Yes/No	57	99.9	99.9	99.7	62.5	99.9	99.9	99.9
Temporal Comparison Yes/No	52.2	100	99.9	100	59.4	99.7	100	99.9
Numerical Superlatives	37.3	96.3	96.2	95.9	67.8	96	96.6	96.4
Temporal Superlatives	33.6	96.6	97.5	97	59.6	97.5	97.8	97.5
Arithmetic Superlatives	42.4	98.2	98.4	97.9	56.6	98.4	98.9	97.6
Arithmetic Addition	7.1	90.4	90.9	91.8	11.8	89.7	91.3	91.1
Counting	46.5	96.8	97.7	98.6	56.1	95.1	97.6	97.7
Date Difference	11.1	92.1	94.3	95.0	11.2	90.7	93.7	94.7

Table 11: F_1 for every task in D_{syn} for T5 and PReasM on the held-out evaluation set.