FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Ekaterina Garanina

Table-to-Text Generation via Logical Forms

Institute of Formal and Applied Linguistics

Supervisor of the master thesis:	Mgr. et Mgr. Ondřej Dušek, Ph.D.
Study programme:	Computer Science
Study branch:	Language Technologies and Computational Linguistics

Prague 2024

First and foremost, I would like to express my gratitude to my supervisors, Prof. Ondřej Dušek and Prof. Dr. Gertjan van Noord, for their support, guidance, and invaluable advice throughout the entire journey of creating this thesis.

I am also grateful to the LCT coordinators, Prof. Tommaso Caselli at RUG and Prof. Markéta Lopatková at Charles University, for their tremendous efforts in managing this double-degree program and resolving all planning issues that I encountered. My sincere thanks also go to the central LCT administration for ensuring the smooth functioning of this program for myself and my classmates from all around the world.

I would like to thank my friends Zhenya, Anya, Kirill, Kostya, Liza, Jacobo, and Toni for their continuous support and readiness to help.

Completing this thesis would not have been possible without the support of my family. I am very grateful to my parents who, despite being thousands of kilometers away, did their best to provide their love and care. And my deepest gratitude goes to my husband Nikolay, who encouraged me to start and complete this Master's program and who has always been there for me no matter the circumstances.

Lastly, I would like to thank the Erasmus+ Programme of the EU for granting me this life-changing opportunity and awarding me the Erasmus Mundus Scholarship.

Title: Table-to-Text Generation via Logical Forms

Author: Ekaterina Garanina

Institute: Institute of Formal and Applied Linguistics

Supervisor: Mgr. et Mgr. Ondřej Dušek, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Table-to-text generation, a subtask of Natural Language Generation (NLG), involves generating coherent and faithful text from structured tables. Modern approaches to this task have overcome language fluency problem but still struggle with generation fidelity, especially in cases where reasoning is required. This thesis focuses on Logical Table-to-Text (LT2T) generation, addressing fidelity, as all previous studies do, but also examining a yet underexplored problem of processing potentially long tables. We investigate the possibility to select the relevant content without using the entire table, as well as the role of symbolic reasoning in enhancing statement fidelity. Our pipeline-based system generates a template of an intermediate logical form from table aggregation and fills the template using a symbolic approach. It outperforms previous works in fidelity and produces natural outputs, as assessed through automatic metrics and human evaluation.

Keywords: natural language processing, natural language generation, table-to-text generation, reasoning, content planning

Contents

1 Introduction

Table-to-text generation is a subtask of Natural Language Generation (NLG) that aims at generating text in natural language from structured tables. There are multiple varieties of this task differing in the domain, table structure, intermediate annotation, and output length (Wiseman et al., 2017; Parikh et al., 2020; Cheng et al., 2022; Moosavi et al., 2021; Lebret et al., 2016; Chen et al., 2020a,c).

The approaches to solving this task also differ and depend substantially on the dataset. Modern solutions include end-to-end training, pre-training models, or pipeline-based systems. At the today's age of large pre-trained models, the systems do not struggle with language fluency anymore, but the major issue is the generation fidelity, i.e. preservation of the correct factual and numerical information in the resulting text (Ji et al., 2022). Another challenge is generating statements involving logical and numerical inference, as modern sequence-to-sequence models lack the reasoning capabilities (Liu et al., 2022a).

This thesis focuses on logical table-to-text (LT2T) generation. All recent works address the fidelity issue using fully neural (Nan et al., 2022; Liu et al., 2022a; Zhao et al., 2022) or partially symbolic (Saha et al., 2022; Zhao et al., 2023a) approaches. Several studies (Perlitz et al., 2023; Zhao et al., 2023b) explore the opportunities to increase the diversity of the outputs. Apart from that, we observe a problem reported but not tackled by previous research, namely, the table length. Long tables contain too much excessive information not required for generation, and they undergo truncation because of the limited context length in the modern language models. In this study, we focus on the questions of fidelity and content selection and create a system that is not dependent on the number of rows in a table.

1.1 Research Questions

We pose two research questions:

1. Is it possible to select relevant content and logical or numerical operations avoiding using the whole table, so that longer tables can be processed?
2. Does symbolic reasoning help to make the statements more faithful to the table?

Our practical goal is to create a pipeline-based system that takes a whole table as input and 1) uses aggregated table information to generate intermediate formal representation templates, 2) fills the template using a symbolic algorithm, 3) generates text from the resulting formal representation. We mainly focus on the first two steps of the pipeline, i.e. content selection and symbolic reasoning, while using a basic model for the third step. Figure 6.1 in Chapter 6 depicts an example of a pipeline execution.

Our main finding is that the described system performs well, being superior in fidelity to other systems according to the automatic metric and producing interesting outputs based on the human evaluation. Therefore, our answer to both research questions is positive. We release the code, the models, and the outputs[1]. We discuss our contributions in more detail in Chapter 8.

1.2 Thesis Structure

This thesis is constructed as follows: in Chapter 2, we provide a brief overview of fundamental notions this work builds on. In Chapter 3, we analyze recent studies, as well as the data and metrics for logical table-to-text generation. We conduct a deeper analysis of the data in Chapter 4. In Chapter 5 we present our baseline, and Chapter 6 contains a full description of our system with the intermediate experiments. We discuss the evaluation and performance of our system in Chapter 7. Finally, our contributions and possible future work directions are outlined in Chapter 8.

[1] `https://github.com/kategerasimenko/LT2T`, also see Appendix A.1.

2 Theoretical Background

In this section, we outline fundamental notions and the background our work builds on. In Section 2.1, we describe the general formulation of the task of the natural language generation, Section 2.2 contains a brief description of a Transformer architecture, and Section 2.3 describes a pre-trained encoder-decoder model that we employ in our work.

2.1 Natural Language Generation

Natural language generation (NLG) is the task of automatically generating coherent texts using a non-linguistic or textual representation of information as input (Reiter and Dale, 2000). The generated texts should correspond to the initial task and remain grammatical, fluent, natural, and faithful to the input.

Common examples of NLG tasks are machine translation and automatic summarization. One of the challenging subtasks of NLG is Data-to-Text (D2T) generation, which generates text in natural language based on non-linguistic input. The concept of D2T is quite broad, including, but not limited to, graph-to-text (Nan et al., 2021), chart-to-text (Obeid and Hoque, 2020), table-to-text (Parikh et al., 2020) tasks.

2.1.1 Table-To-Text Generation

In this study, we focus on table-to-text generation, which aims to produce statements that highlight crucial table elements, either reporting mostly surface-level values (Parikh et al., 2020; Cheng et al., 2022; Lebret et al., 2016) or involving rich logical reasoning (Chen et al., 2020a,c). With the recent development of large language models with astonishing generation capabilities (Radford et al., 2019; Lewis et al., 2020; Raffel et al., 2020; Touvron et al., 2023), language fluency is a solved problem for high-resource languages, but processing complex data, reasoning, and revealing non-obvious relations in the input is still a challenge (Zhao et al., 2022; Liu et al., 2022a). In this work, we address this issue and focus on logical table-to-text generation. Further background on this task is provided in Chapter 3.

2.2 Transformer

Transformer (Vaswani et al., 2017) is the architecture used by state-of-the-art systems for multiple NLP tasks (Wolf et al., 2020). As opposed to the previous best architecture for processing sequences, RNNs (Rumelhart and McClelland, 1987), Transformers rely solely on attention mechanisms, discarding recurrence and convolutions, thus making the model faster, more scalable and more powerful. The general Transformer architecture is depicted in Figure 2.1. The whole Transformer consists of an encoder and a decoder, which allows to use it for sequence-to-sequence tasks, such as the table-to-text task considered in this work.

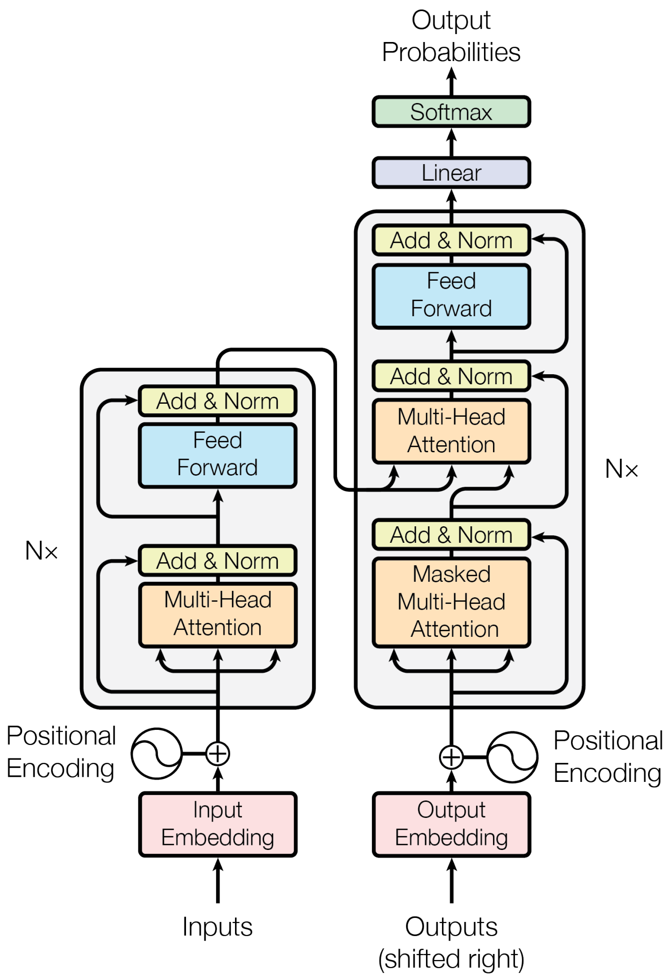


Figure 2.1: Transformer architecture. Taken from Vaswani et al. (2017).

2.3 Text-to-Text Transfer Transformer

Text-to-Text Transfer Transformer (Raffel et al., 2020), abbreviated as T5, is a model pre-trained in the framework which casts multiple language tasks (e.g., translation, summarization, sentiment analysis, sentence similarity estimation) into the text-to-text formulation. This allows to train one model in a multi-task setup without any additional heads and other changes to the original architecture.

Figure 2.2 shows the principle of the text-to-text framework used to pre-train T5 models.

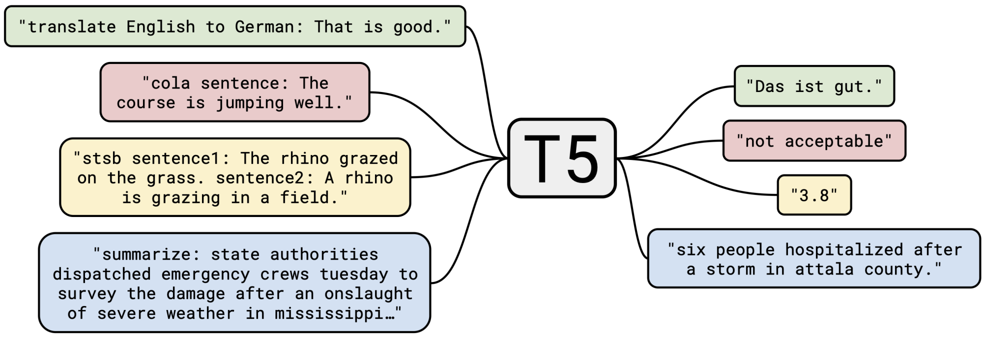

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Figure 2.2: Text-to-text framework used for T5 pre-training. Taken from Raffel et al. (2020).

This approach showed state of the art performance in multiple tasks at the moment of the publication, and the pre-trained models of different sizes were released for usage and fine-tuning.

3 Related Work

In this section, we give an overview of the recent advances in logical table-to-text (LT2T) generation. The interest to this task has been growing since the release of LogicNLG dataset (Chen et al., 2020a). LT2T generation is aimed at generating natural language statements that can be logically entailed by the facts in an open-domain table. Therefore, LT2T implies reasoning and producing sentences that are consistent both linguistically and logically with the input table.

The task is extremely challenging because, opposed to the task of reporting the information from pre-defined table cells, as in ToTTo dataset (Parikh et al., 2020), the system needs to capture subtle non-trivial relations and include complex reasoning into the statements. There are several challenges that the current LT2T research addresses:

1. fidelity: language models struggle with logical and numerical reasoning, as shown by multiple works (Chen et al., 2020a; Liu et al., 2022a; Zhao et al., 2022).
2. diversity: the space of possible table descriptions is exponentially large, and it is required for the system to produce a valid, diverse, and interesting set of statements covering different parts of a table (Chen et al., 2020c; Perlitz et al., 2023; Zhao et al., 2023b).

Another issue is the table size which further complicates content selection and depends on a technical restriction of the maximum model input length. A related study on table reasoning (Chen, 2023) reports a dramatic drop in performance with the increasing table size. However, this challenge is underexplored in current works on LT2T, which mostly use provided pre-selected columns (Liu et al., 2022a; Zhao et al., 2022; Perlitz et al., 2023; Zhao et al., 2023b) and employ table truncation strategies (Zhao et al., 2023c).

We briefly describe the main datasets (Section 3.1), discuss available evaluation metrics (Section 3.2), and outline the recent research in the LT2T field (Section 3.3).

3.1 Datasets

Several table-to-text datasets are specifically tailored for numerical and logical reasoning. They differ in size and in domain.

LogicNLG is the largest dataset of its kind and features very rich logical inference on open-domain tables. NumericNLG (Suadaa et al., 2021) and SciGen (Moosavi et al., 2021) contain numerical tables with the statements involving reasoning in the scientific domain. Logic2Text (Chen et al., 2020c) consists of tables, sentences, and corresponding logical forms as intermediate formal representations.

In the current study, we extensively use Logic2Text for development, and test our system on LogicNLG. We provide more in-depth analysis of these two datasets in Chapter 4.

3.2 Evaluation Metrics

Previous works use two general types of metrics: reference-based, which compare a statement to the set of references, and table-based, which evaluate sentences based on the input table to measure fidelity. The first type is a traditional way to evaluate a sequence-to-sequence model, e.g., in machine translation. The second evaluation type is aimed at measuring logical fidelity of the statements and requires a table as input.

3.2.1 Reference-Based Metrics

A large variety of reference-based metrics is used for different sequence-to-sequence tasks: BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), and others. Works on LT2T generation employ BLEU score most frequently.

BLEU (bilingual evaluation understudy) is a token-based metric, which compares n-gram overlap between a candidate and a set of references. BLEU score ranges from 0 to 1, where 1 indicates a perfect match with the reference translation(s) and 0 indicates no overlap at all. It is computed with the following formula:

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

In this formula:

- N is the maximum order of n-grams;
- w_n is the weight of the corresponding n-gram (usually uniform);
- p_n is the modified n-gram precision, restricted by n-gram counts in the references;
- BP is the brevity penalty, aimed at discouraging shorter candidate sequences and calculated as follows:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}$$

  where c is the candidate text length and r is the length of the closest matching reference translation.

Despite being extremely widely used, BLEU is criticized for the heavy dependence on tokenization strategy (Post, 2018), lack of recall measurement (Banerjee and Lavie, 2005), and relying only on a surface realization without capturing the actual meaning (Sellam et al., 2020).

BLEU score is usually computed on the whole candidate sentence set, where p_n is calculated based on the summed counts from all sentences in the set. Another convention for standard tasks like machine translation is using one standartized implementation, namely, SacreBLEU (Post, 2018), which features n-gram order of 4 and a high-level interface for running the evaluation. However, in LT2T, related studies follow Chen et al. (2020a) and measure averaged sentence-level BLEU-1/2/3 in NLTK (Bird et al., 2009) implementation.

3.2.2 Table-Based Metrics

Since LT2T implies reporting non-obvious complex facts from the table, rule-based surface-matching metrics cannot be used to get an adequate fidelity estimation. Therefore, the common practice in the field is to use model-based metrics that rely either on automatic semantic parsing (SP) or on models that perform Natural Language Inference (NLI) on the tables, i.e., predict whether the statement is entailed or refuted by the table.

We consider four metrics: SP-acc and NLI-acc proposed by Chen et al. (2020a), and TaPas-acc and TaPEx-acc based on the TaPas (Herzig et al., 2020) and TaPEx (Liu et al., 2022b) models accordingly. NLI models are trained on TabFact dataset (Chen et al., 2020b), which contains both positive and negative examples of statement entailment.

1. SP-acc (Chen et al., 2020a) is based on weakly-supervised semantic parsing. It extracts the meaning representation from the generated sentence and executes it against the table to verify its correctness. As found by Liu et al. (2022a), the parsing algorithm for SP-Acc often generates irrelevant logical programs for the sentences, which makes the evaluation unreliable.

2. NLI-acc (Chen et al., 2020a) is based on TableBERT (Chen et al., 2020b), which achieves relatively low accuracy of 65.1% on the TabFact dataset and is overly positive about the predictions, as reported by Liu et al. (2022a).

3. TaPas-acc employs TaPas model (Herzig et al., 2020; Eisenschlos et al., 2020). The model is a derivation of BERT with additional embeddings to encode the table structure, and it achieves quality improvement by the additional pre-training on two auxiliary tasks involving synthetic and counterfactual statements. TaPas shows accuracy of 81% on the TabFact test set.

4. TaPEx-acc uses TaPEx model (Liu et al., 2022b) which is a further improvement on TabFact. TaPEx uses BART as a backbone model and also benefits from additional pre-training, learning a neural SQL executor on a diverse large-scale synthetic corpus. It performs with the 84.2% accuracy on TabFact.

Overall, the studies use SP-acc and NLI-acc for comparability with other works. As seen from the reported performance accuracy on TabFact, TaPEx-acc is considered to be the most reliable metric for LT2T so far, although it is not used by everyone. In our work, we use TaPas and TaPEx accuracy and do not rely on the first two metrics because of the lower quality.

3.3 Existing Approaches

In this section, we describe and analyze works based on Logic2Text or LogicNLG. We summarize the research directions and list concrete studies in more detail.

Overall, all approaches to LT2T utilize pre-trained language models but even larger ones are shown to struggle with the task. Therefore, the proposed methods include other additions and improvements. Related studies suggest using additional components in the model (Chen et al., 2021; Nan et al., 2022), logic-related pre-training (Liu et al., 2022a; Zhao et al., 2022), neuro-symbolic approaches with reasoning done by external programs (Saha et al., 2022; Zhao et al., 2023a,b), and controlling the generation explicitly (Perlitz et al., 2023; Zhao et al., 2023b). Among the most recent works, Zhao et al. (2023c) analyze large language model (LLM) prompting and report impressive results.

Chen et al. (2020a), apart from releasing LogicNLG dataset and novel table-based metrics, consider various approaches to training the model on this data. They show that the task is extremely challenging by surveying different architectures (LSTM, Transformer), pre-trained models (GPT-2), training strategies (RL, Adversarial Training, Coarse-to-Fine), and concluding that all models lack fidelity based on human evaluation. Subsequent works manage to improve the performance on LogicNLG dataset.

Chen et al. (2021) argue that baseline models learn to capture surface-level spurions correlations between the table and the sentence instead of correct causal relations. To mitigate this problem, they propose a de-confounded variational encoder-decoder (DCVED) based on causal intervention, back-prediction process, and a table-text selector, and achieve state of the art results at that moment.

Nan et al. (2022) argue that entity retrieval capability of D2T systems is one of the primary faithfulness factor. They propose training a system both as a generator and a faithfulness discriminator with additional replacement detection and unlikelihood learning tasks. They experiment with multiple table-to-text datasets and do not address logical table-to-text generation specifically, but the proposed system surpasses the baselines and previous works on LogicNLG according to the automatic metric.

Liu et al. (2022a) and Zhao et al. (2022) follow the general trend of further pre-training, which is successfully applied in other table-related NLP tasks as well (Andrejczuk et al., 2022; Herzig et al., 2020; Liu et al., 2022b). Liu et al. (2022a) show that table-to-logic pre-training on a large synthetic dataset greatly improves the fidelity on downstream table-to-text tasks. Zhao et al. (2022) construct a template-based dataset of questions on tables, and use question answering via generation as a pre-training task. This approach outperformed all previous studies by a considerable margin.

Saha et al. (2022) propose a neuro-symbolic modular system with multi-step reasoning. They develop several modules and a grammar for their combination, using neural networks for constructing coherent text and symbolic algorithms for reasoning to ensure fidelity. They focus on a few-shot setting, showing that neuro-symbolic approach outperforms other few-shot techniques such as direct and chain-of-thought prompting.

Perlitz et al. (2023) and Zhao et al. (2023b) address the problem of output diversity along with fidelity. Perlitz et al. (2023) use logic types, i.e., logical or numerical operations required to generate a statement, to control the generation. The system accepts the explicit logic types, which can be sampled randomly, and thus is guided to produce more diverse outputs keeping fidelity. Zhao et al. (2023b) utilize logical forms to select the content and relevant operations and conduct reasoning, thus controlling logical table-to-text generation and preserving the fidelity of the output.

Zhao et al. (2023a) rely on a neuro-symbolic approach, generating a statement template and reasoning out concrete entities from a table by a dependency-aware symbolic reasoning framework based on table-compatible programming language. They show a further improvement in output fidelity, though not addressing the diversity problem.

Zhao et al. (2023c) explore the most recent trend of LLM prompting and find that querying GPT-* models outperforms all existing solutions by a large margin, achieving impressive fidelity. They also show that LLMs can also be useful for evaluation and feedback on other models' output. It is important to note that this analysis relies only on 500 examples due to technical constraints.

Overall, the field is rapidly developing, and a huge improvement in generation quality has been achieved since the first dataset publication. Nevertheless, the challenges of fidelity, diversity, and controlled generation remain to be highly relevant.

4 Data

This research utilizes two datasets: LogicNLG (Chen et al., 2020a) and Logic2Text (Chen et al., 2020c). These datasets consist of tables coupled with corresponding statements which include reasoning about the content of the tables. These tables are sourced from the TabFact dataset (Chen et al., 2020b), which is itself based on WikiTables (Bhagavatula et al., 2013). TabFact is modified by filtering out tables that are overly complicated or large (for example, those with multiple rows or columns, or LaTeX symbols), resulting in relatively clean tables of flat layout and reasonable size. Each table contains a one-row header and a title.

In handling Logic2Text, we had to exclude all training tables already present in the LogicNLG development and test sets. In further dataset analysis, we use an updated version of the dataset.

Each table carries a unique identifier, referred to as `table_id` (for Logic2Text, this is the final part of the `url` attribute). For ease of reference throughout this text, we use both the `table_id` and a combination of split and example number from the original dataset[1]. The dataset statistics are detailed in Table 4.1, and the train, development, and test splits are displayed in Table 4.2.

	LogicNLG	**Logic2Text**
Tables	7392	4855
References	37015	9327
Refs per table	5.007	1.921
Max rows	49	21
Max columns	26	25
Ref length	14.227	16.908

Table 4.1: Dataset statistics.

	LogicNLG		**Logic2Text**	
	Tables	**References**	**Tables**	**References**
train	5682	28450	3855	7140
dev	848	4260	500	1095
test	862	4305	500	1092

Table 4.2: Dataset split statistics.

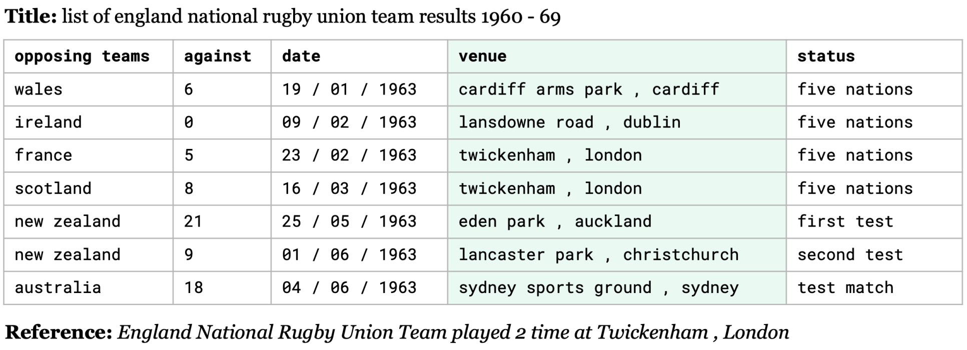
[1]We use the Tabgenie toolkit (Kasner et al., 2023) for dataset navigation and processing.

4.1 LogicNLG

LogicNLG is a direct derivative of TabFact dataset. The authors of LogicNLG selected true statements from TabFact that belong to the category of higher complexity, which denotes reasoning in TabFact annotations.

Additionally, the authors conducted a thorough preprocessing of the raw data. An automatic entity linking system was employed to identify the relevant columns for each corresponding reference. This data was included as column selection and a reference template. While not explicitly mentioned in the dataset description, references were also also preprocessed to emphasize column mentions and values.

An example of a resulting table-reference pair is depicted in Figure 4.1.



Title: list of england national rugby union team results 1960 - 69

opposing teams	against	date	venue	status
wales	6	19 / 01 / 1963	cardiff arms park , cardiff	five nations
ireland	0	09 / 02 / 1963	lansdowne road , dublin	five nations
france	5	23 / 02 / 1963	twickenham , london	five nations
scotland	8	16 / 03 / 1963	twickenham , london	five nations
new zealand	21	25 / 05 / 1963	eden park , auckland	first test
new zealand	9	01 / 06 / 1963	lancaster park , christchurch	second test
australia	18	04 / 06 / 1963	sydney sports ground , sydney	test match

Reference: *England National Rugby Union Team played 2 time at Twickenham , London*

Figure 4.1: Table with the title and column selection and corresponding reference from LogicNLG (train 6, id `2-18179114-4`).

Nevertheless, we identified two issues with the preprocessed references:

1. The resulting reference is not formulated in natural language, potentially affecting the performance of models that have been pre-trained on written texts.
2. Since the end-to-end evaluation aims to use a model trained on TabFact, the input must mirror TabFact training data, which is in natural language. Therefore, this preprocessing introduces an inconsistency in the input, possibly leading to lower scores.

Considering these issues with preprocessed references, we have reverted them to match the original TabFact statements. We still use original LogicNLG data to compare some of our experiments with the previous works.

4.2 Logic2Text

In contrast to LogicNLG, Logic2Text is less related to TabFact since the references for Logic2Text were obtained independently. Beyond tables and references, the dataset includes corresponding intermediate formal representations, denoted as "logical forms" (LFs). The syntax of these LFs is custom yet formalized, permitting parsing into a nested structure and execution on a table.

References were collected by crowdsourcing, where the annotators were provided with logical operators to utilize in sentence construction. Afterwards, full logical forms were created as a separate task within a conversational setting, with each function having specifically designed questions. To confirm the correctness of the resulting forms, the logical forms were then executed against the table.

Figure 4.2 depicts a table from Logic2Text dataset with the corresponding logical forms and references.

Title: wake forest demon deacons football , 1980 - 89

date	opponent	location	result	attendance
09 / 12 / 1987	richmond	groves stadium winston - salem , nc	w 24 - 0	14250
09 / 19 / 1987	north carolina state	groves stadium winston - salem , nc	w 21 - 3	23600
09 / 26 / 1987	appalachian state	groves stadium winston - salem , nc	w 16 - 12	33400
10 / 01 / 1987	army	michie stadium west point , ny	w 17 - 13	36690
10 / 10 / 1987	north carolina	kenan memorial stadium chapel hill , nc	w 22 - 14	50000
10 / 17 / 1987	maryland	groves stadium winston - salem , nc	l 0 - 14	25175
10 / 24 / 1987	virginia	scott stadium charlottesville , va	l 21 - 35	32500
10 / 31 / 1987	14 clemson	memorial stadium clemson , sc	l 17 - 31	69711
11 / 07 / 1987	duke	groves stadium winston - salem , nc	w 30 - 27	23500
11 / 14 / 1987	14 south carolina	groves stadium winston - salem , nc	l 0 - 30	34720
11 / 21 / 1987	georgia tech	grant field atlanta , ga	w 33 - 6	21114

Logical forms	**References**
most_eq { all_rows ; result ; w }	*most of the games resulted in wins for the wake forest demon deacons .*
round_eq { avg { all_rows ; attendance } ; 33151 }	*the average attendance for the wake forest demon deacons games was 33151 .*
eq { hop { argmax { all_rows ; attendance } ; location } ; memorial stadium clemson , sc }	*the game played at memorial stadium clemson was the highest attended game .*

Figure 4.2: Table, logical forms, and references from Logic2Text (train 6995, id `2-15531181-15`).

Logical forms consist of nested functions which take a specified set of arguments. There are 39 functions in the dataset. Function definitions are given in Table 4.3, and the corresponding examples are provided in Table A.1 in Appendix A.2. Figure 4.3 contains two LF examples with the explanation of an LF structure.

```
round_eq {  # rough equality between two values
        avg { all_rows ; votes } ;  # average of `votes` column across all rows
        13004
}


and {  # boolean AND between two boolean arguments
        only {  # boolean: whether there is only one value selected
                filter_eq { all_rows ; weapon ; razor boomerang }  # filtering by `weapon` column by `razor boomerang` value
        } ;
        eq {  # equality between two arguments: result of `hop` and `hunter harris`
                hop {  # from the row in the first argument, take value in the column `eagle riders`
                        filter_eq { all_rows ; weapon ; razor boomerang } ;  # same filtering as in `only` function
                        eagle riders
                } ;
                hunter harris
        }
}
```

Figure 4.3: Two examples of logical forms. The first one is simple and calculates the average (dev 2, `2-12890300-1`). The second one checks the value uniqueness and reports the value from another column; it requires the filtering consistency across different LF arguments (dev 0, `2-17464729-1`).

Name	Arguments	Output	Description
count	view	number	returns the number of rows in the view
only	view	bool	returns whether there is exactly one row in the view
hop	row, header	value	returns the value under the header column of the row
and	bool, bool	bool	returns the boolean operation result of two arguments
max/min/avg/sum	view, header	number	returns the max/min/average/sum of the values under the header column
nth_max/nth_min	view, header, integer	number	returns the n-th max/n-th min of the values under the header column
argmax/argmin	view, header	row	returns the row with the max/min value in header column
nth_argmax/nth_argmin	view, header, integer	row	returns the row with the n-th max/min value in header column
eq/not_eq	value, value	bool	returns if the two arguments are equal
round_eq	value, value	bool	returns if the two arguments are roughly equal under certain tolerance
greater/less	value, value	bool	returns if argument 1 is greater/less than argument 2
diff	value, value	value	returns the difference between two arguments
filter_eq/not_eq	view, header, value	view	returns the subview whose values under the header column is equal/not equal to argument 3
filter_greater/less	view, header, value	view	returns the subview whose values under the header column is greater/less than argument 3
filter_greater_eq /less_eq	view, header, value	view	returns the subview whose values under the header column is greater/less or equal than argument 3
filter_all	view, header	view	returns the view itself for the case of describing the whole table
all_eq/not_eq	view, header, value	bool	returns whether all the values under the header column are equal/not equal to argument 3
all_greater/less	view, header, value	bool	returns whether all the values under the header column are greater/less than argument 3
all_greater_eq/less_eq	view, header, value	bool	returns whether all the values under the header column are greater/less or equal to argument 3
most_eq/not_eq	view, header, value	bool	returns whether most of the values under the header column are equal/not equal to argument 3
most_greater/less	view, header, value	bool	returns whether most of the values under the header column are greater/less than argument 3
most_greater_eq/less_eq	view, header, value	bool	returns whether most of the values under the header column are greater/less or equal to argument 3

Table 4.3: Function definitions. Taken from Chen et al. (2020c) with minor changes.

Further analysis of logical forms shows that the dataset relies on certain LF patterns, which we will subsequently call "structures". We derive these structures using heuristics by replacing values and headers with placeholders `X` and `Y`. As an example, consider the following LF and its corresponding structure:

```
most_eq { all_rows ; played ; 114 }
most_eq { all_rows ; Y ; X }
```

Across the entire Logic2Text dataset, there are 191 distinct structures. Table 4.4 displays the top five structures along with their absolute frequency in the dataset. The data shows that the most prevalent LF structure constitutes roughly 12% (1154 / 9327) of the total dataset, while these five structures combined account for approximately 47% of the dataset.

structure	**count**
`eq { count { filter_eq { all_rows ; Y ; X } } ; X }`	1154
`and { only { filter_eq { all_rows ; Y ; X } } ; eq { hop { filter_eq { all_rows ; Y ; X } ; Y } ; X } }`	895
`most_eq { all_rows ; Y ; X }`	854
`round_eq { avg { all_rows ; Y } ; X }`	739
`eq { hop { argmax { all_rows ; Y } ; Y } ; X }`	727

Table 4.4: Top 5 LF structures and their counts in Logic2Text dataset.

The function distribution in the data also indicates a considerable imbalance. Figure 4.4 illustrates the proportions of different function occurrences in the data (multiple occurrences of a function within a single LF are counted individually). The functions `filter_eq`, `eq`, and `hop` can be considered more "technical" in

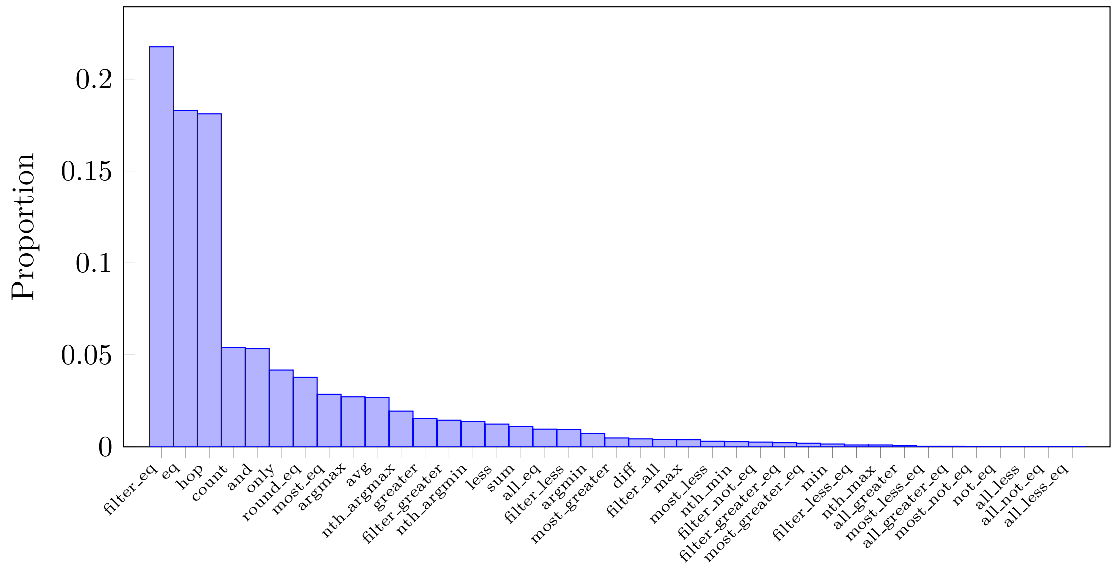


Figure 4.4: Function distribution across Logic2Text dataset.

nature, and their presence in many LFs can be attributed to their functional roles. Nevertheless, quite a lot of functions form a long infrequent tail, with `max / min` and `diff` being the most unexpected.

5 Baseline

In this section, we outline our baseline system and compare it to the baselines from previous studies to ensure its validity. Moreover, we perform a series of experiments to enhance the robustness of our baseline and explore the challenges of the task. For evaluation, we employ the metrics described in Section 3.2, specifically, TAPEX-acc, TAPAS-acc, and the BLEU score. For the BLEU calculation, we calculate SacreBLEU as a standard sequence-to-sequence metric, as well as BLEU-3 in order to be in line with the previous research on LogicNLG. In this section, we follow the evaluation setup of the previous works, but we use a slightly different strategy in our final evaluation (described in Section 7.2).

First, we have rerun the training of the baseline system from Liu et al. (2022a) and achieved results comparable to the reported ones. Nonetheless, even the column selection does not completely resolve the technical issue of handling large inputs: certain inputs get truncated due to the large size of some tables. Moreover, the input in this system includes certain pre-computed values, which we avoid in our baseline, keeping it as basic as possible.

We develop a sequence-to-sequence baseline that takes selected columns as input and produces generated statements as output. Our baseline is a simple end-to-end system which is based on fine-tuning a pre-trained LM. We conduct several experiments on `t5-base` model, and all parameters that we fix during training are listed in Appendix A.4.1. We discuss our experiments in the subsequent sections.

5.1 Input Formats

As a side experiment, we explore different input formats. All our representations are designed to be more concise to mitigate the issue of truncation, but each option encodes information slightly differently. These formats are quite different from Liu et al. (2022a) where the input is in XML format. The formats we examine are as follows:

- **Markers**: cells are listed with their headers, with special marker tokens denoting row start, cell, and header cell:

  ```
  <T> oleh protasov <R> 1 <H> goal <C> 1 <H> date <C> 15
  may 1984 <H> competition <C> friendly <R> 2 <H> goal <C>
  2 <H> date <C> 2 june 1984 <H> competition <C> friendly
  ...
  ```

- **NL: natural language**: cells are listed with their headers in a concise but natural language-like format, without special tokens but with the usual punctuation:

```
Title: oleh protasov. Row, goal: 1, date: 15 may 1984,
competition: friendly. Row, goal: 2, date: 2 june 1984,
competition: friendly. ...
```

- **NL-idx: natural language with row indices**: the representation is the same as NL but with the addition of row indices:

```
Title: oleh protasov. Row: 1, goal: 1, date: 15 may
1984, competition: friendly. Row: 2, goal: 2, date: 2
june 1984, competition: friendly. ...
```

5.2 Original LogicNLG Outputs

As discussed in Section 4.1, for the proper application of the TaPEx-acc and TaPas-acc metrics, the system should produce statements similar to the TabFact dataset which both models were trained on. However, PLOG models, against which our system is compared, are trained on the original outputs of LogicNLG.

Thus, we train two types of baseline models: initially, we select the most effective input format based on the corrected outputs, then we train another model using the same input format, but with the original LogicNLG outputs, in order to compare our baseline to that from Liu et al. (2022a). Additionally, this provides us with a rough estimation of the variation in TaPEx-acc and TaPas-acc that could be expected from correcting the outputs.

5.3 Results

Table 5.1 illustrates the performance of models trained on different input formats.

	train	dev				test			
	trunc	BLEU-3	SacreBLEU	TaPEx	TaPas	BLEU-3	SacreBLEU	TaPEx	TaPas
Markers	13.5	18.1	17.5	60.3	64.7	18.7	17.5	60.9	63.1
NL	**12.8**	**18.9**	**17.8**	**63.8**	**67.4**	**19.5**	**18.3**	62.2	65.3
NL-idx	14.0	18.0	17.4	62.7	67.3	19.1	**18.3**	**62.8**	**66.9**

Table 5.1: Comparison of different input formats on LogicNLG with TabFact outputs. Metrics are described in Section 3.2; *trunc* stands for the percentage of truncated inputs.

NL encoding strategy shows superior results on the development set but performs worse than NL-idx input format with respect to fidelity metric. We rely on the test set results and compare the model trained on the NL-idx input format and

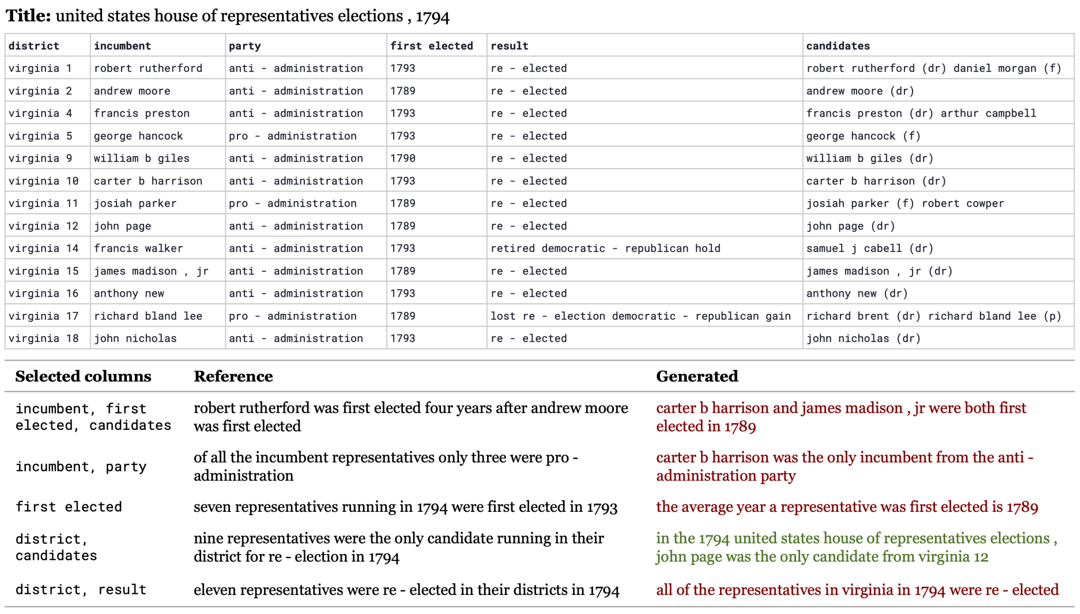
	train	dev			test		
	trunc	BLEU-3	TaPEx	TaPas	BLEU-3	TaPEx	TaPas
Baseline from Liu et al. (2022a)	32.2	**19.1**	55.9	58.4	**19.9**	55.5	56.1
T5-base NL-idx	**14.0**	18.5	**61.1**	**62.0**	19.2	**59.4**	**60.9**

Table 5.2: Comparison of a `t5-base` baseline from Liu et al. (2022a) and our baseline on LogicNLG with original outputs. *trunc* stands for the percentage of truncated inputs.

original LogicNLG outputs with the baseline model from Liu et al. (2022a). Table 5.2 reveals that:

- The PLOG input format truncates a considerable amount of inputs, which could potentially mitigate the model's performance.
- The baseline by Liu et al. (2022a) surpasses our model in the BLEU-3 metric, while our model achieves higher fidelity scores. This discrepancy could be due to our focus on maximizing the TaPEx-acc score, whereas PLOG employs the BLEU-3 metric for model selection.
- The difference in TaPEx-acc score between models trained on original and corrected outputs is around 1.5 points for development sets and approximately 3.5 points for test sets, favoring the corrected version. The same holds for TaPas-acc, where the difference is more than 5 points for both sets. This proves our hypothesis concerning the importance of data consistency.

We have examined a small number of outputs produced by the baseline and observed relatively low fidelity. Figure 5.1 displays the table from LogicNLG and the generated statements.



Title: united states house of representatives elections , 1794

district	incumbent	party	first elected	result	candidates
virginia 1	robert rutherford	anti - administration	1793	re - elected	robert rutherford (dr) daniel morgan (f)
virginia 2	andrew moore	anti - administration	1789	re - elected	andrew moore (dr)
virginia 4	francis preston	anti - administration	1793	re - elected	francis preston (dr) arthur campbell
virginia 5	george hancock	pro - administration	1793	re - elected	george hancock (f)
virginia 9	william b giles	anti - administration	1790	re - elected	william b giles (dr)
virginia 10	carter b harrison	anti - administration	1793	re - elected	carter b harrison (dr)
virginia 11	josiah parker	pro - administration	1789	re - elected	josiah parker (f) robert cowper
virginia 12	john page	anti - administration	1789	re - elected	john page (dr)
virginia 14	francis walker	anti - administration	1793	retired democratic - republican hold	samuel j cabell (dr)
virginia 15	james madison , jr	anti - administration	1789	re - elected	james madison , jr (dr)
virginia 16	anthony new	anti - administration	1793	re - elected	anthony new (dr)
virginia 17	richard bland lee	pro - administration	1789	lost re - election democratic - republican gain	richard brent (dr) richard bland lee (p)
virginia 18	john nicholas	anti - administration	1793	re - elected	john nicholas (dr)

Selected columns	Reference	Generated
`incumbent, first elected, candidates`	robert rutherford was first elected four years after andrew moore was first elected	carter b harrison and james madison , jr were both first elected in 1789
`incumbent, party`	of all the incumbent representatives only three were pro - administration	carter b harrison was the only incumbent from the anti - administration party
`first elected`	seven representatives running in 1794 were first elected in 1793	the average year a representative was first elected is 1789
`district, candidates`	nine representatives were the only candidate running in their district for re - election in 1794	in the 1794 united states house of representatives elections , john page was the only candidate from virginia 12
`district, result`	eleven representatives were re - elected in their districts in 1794	all of the representatives in virginia in 1794 were re - elected

Figure 5.1: Example of a baseline system output (LogicNLG, test 35-39, id `1-2668420-17`). There are 5 dataset entries for the given table with varying column selection (*selected columns*) and the corresponding references. Baseline model uses only selected columns. A correct generation is in green, unfaithful ones are in red.

6 Proposed System

In this section, we describe our pipeline-based system and analyze the performance of individual steps.

6.1 Overview

Our system is a pipeline-based approach that leverages symbolic operations. By breaking the process into multiple steps, we aim to increase the fidelity of generation and offer greater control and interpretability in the generation process.

Specifically, our system consists of four steps:

1. Data preprocessing (Section 6.2). This step prepares the tables for both training and inference. In particular, the columns are parsed according to their identified data types. Column type classification and value parsing are essential for a more meaningful and directed content selection and subsequent symbolic reasoning.

2. Content selection (Section 6.3). This step generates an LF template, which can also be viewed as generation planning. A sequence-to-sequence model is given a more abstract representation of the table rather than the table itself, and it produces a nested structure of operations on appropriate columns, without specific values or results of operation execution. This step aims to achieve two objectives:

   (a) we select relevant content and operations without passing actual values, thus reducing noise in the input and its overall length;

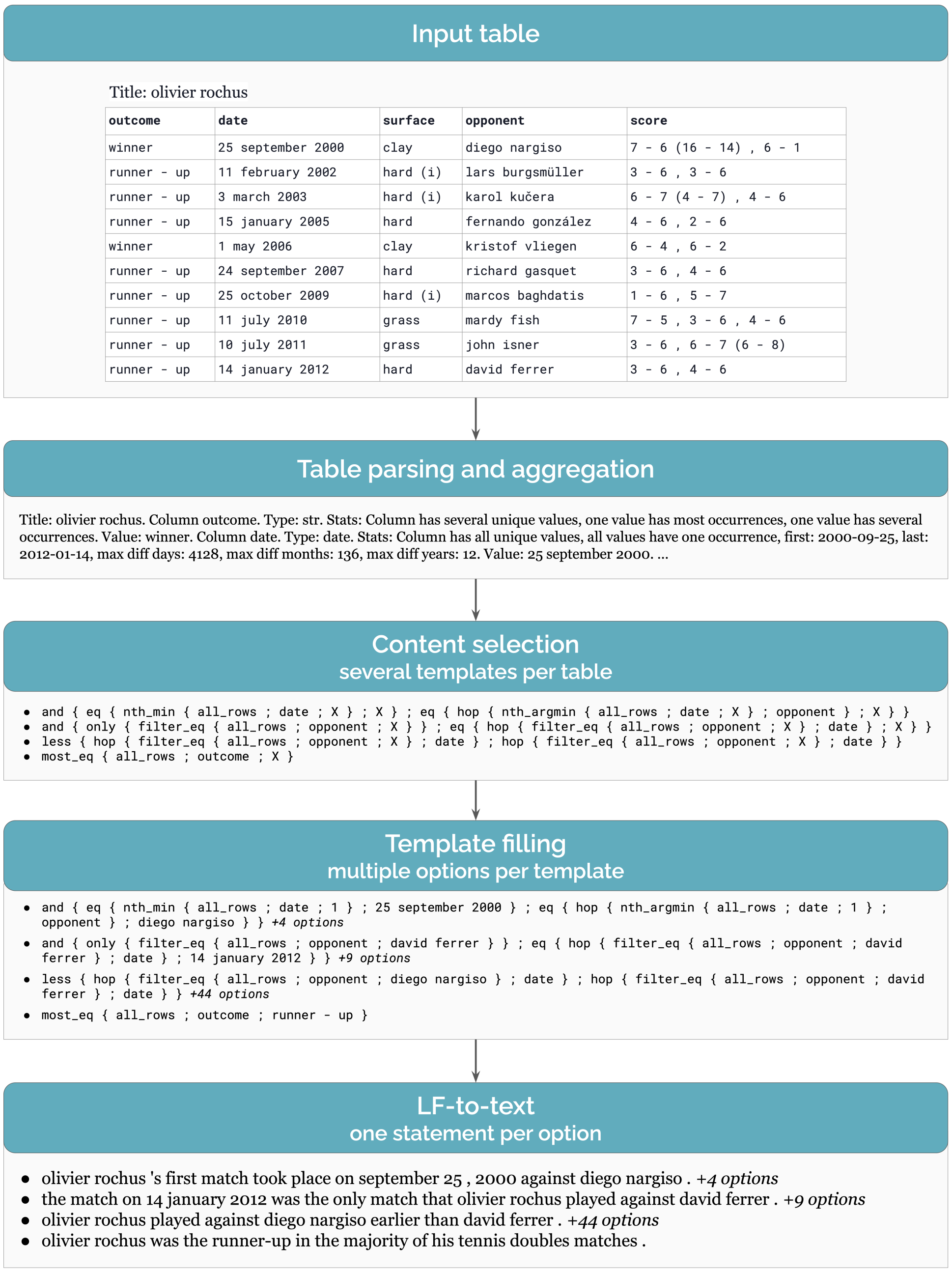
   (b) we do not force the model to copy values and, most importantly, do any kind of reasoning on 2D data.

3. Template filling (Section 6.4). This stage constitutes the symbolic part of our approach. We have developed an algorithm for recursively filling the LF template generated in the previous step. The algorithm selects values for all available functions and expands the option tree, finally producing all possible template filling options for the given template. By applying a symbolic approach in template filling, we minimize reasoning and calculation errors, thereby enhancing the overall fidelity of the system.

4. LF-to-text generation (Section 6.5). This step involves using a sequence-to-sequence model trained in a straightforward manner. We expect the model to be capable of deriving natural language counterparts from function names

and copying the corresponding pre-calculated values. This results in the generation of a fluent yet accurate statement in natural language, which is the ultimate goal of the pipeline.

Figure 6.1 depicts the example of a pipeline executed on one table. In the subsequent sections, we outline all components of the system, evaluate their performance, and provide relevant intermediate experiments. We include the majority of table examples in Appendix A.3.



Title: olivier rochus

outcome	date	surface	opponent	score
winner	25 september 2000	clay	diego nargiso	7 - 6 (16 - 14) , 6 - 1
runner - up	11 february 2002	hard (i)	lars burgsmüller	3 - 6 , 3 - 6
runner - up	3 march 2003	hard (i)	karol kučera	6 - 7 (4 - 7) , 4 - 6
runner - up	15 january 2005	hard	fernando gonzález	4 - 6 , 2 - 6
winner	1 may 2006	clay	kristof vliegen	6 - 4 , 6 - 2
runner - up	24 september 2007	hard	richard gasquet	3 - 6 , 4 - 6
runner - up	25 october 2009	hard (i)	marcos baghdatis	1 - 6 , 5 - 7
runner - up	11 july 2010	grass	mardy fish	7 - 5 , 3 - 6 , 4 - 6
runner - up	10 july 2011	grass	john isner	3 - 6 , 6 - 7 (6 - 8)
runner - up	14 january 2012	hard	david ferrer	3 - 6 , 4 - 6

Title: olivier rochus. Column outcome. Type: str. Stats: Column has several unique values, one value has most occurrences, one value has several occurrences. Value: winner. Column date. Type: date. Stats: Column has all unique values, all values have one occurrence, first: 2000-09-25, last: 2012-01-14, max diff days: 4128, max diff months: 136, max diff years: 12. Value: 25 september 2000. ...

```
• and { eq { nth_min { all_rows ; date ; X } ; X } ; eq { hop { nth_argmin { all_rows ; date ; X } ; opponent } ; X } }
• and { only { filter_eq { all_rows ; opponent ; X } } ; eq { hop { filter_eq { all_rows ; opponent ; X } ; date } ; X } }
• less { hop { filter_eq { all_rows ; opponent ; X } ; date } ; hop { filter_eq { all_rows ; opponent ; X } ; date } }
• most_eq { all_rows ; outcome ; X }
```

```
• and { eq { nth_min { all_rows ; date ; 1 } ; 25 september 2000 } ; eq { hop { nth_argmin { all_rows ; date ; 1 } ;
  opponent } ; diego nargiso } } +4 options
• and { only { filter_eq { all_rows ; opponent ; david ferrer } } ; eq { hop { filter_eq { all_rows ; opponent ; david
  ferrer } ; date } ; 14 january 2012 } } +9 options
• less { hop { filter_eq { all_rows ; opponent ; diego nargiso } ; date } ; hop { filter_eq { all_rows ; opponent ; david
  ferrer } ; date } } +44 options
• most_eq { all_rows ; outcome ; runner - up }
```

- olivier rochus 's first match took place on september 25 , 2000 against diego nargiso . *+4 options*
- the match on 14 january 2012 was the only match that olivier rochus played against david ferrer . *+9 options*
- olivier rochus played against diego nargiso earlier than david ferrer . *+44 options*
- olivier rochus was the runner-up in the majority of his tennis doubles matches .

Figure 6.1: Example of the pipeline execution on one table. LogicNLG, test 150, id `2-1554464-3`.

6.2 Data Preprocessing

To perform symbolic reasoning, it is necessary to determine the data type of each column and parse the table values accordingly, maintaining both the original and processed values.

First, we remove the last row if it contains summations (`total` or `sum` in the first cell of the row, see Figure A.1), in line with previous works. Additionally, we delete all rows that duplicate the header (see Figure A.2). Finalizing first basic preprocessing, we replace all variations of `n/a` representations (e.g., Figure A.3) with Python `None` objects.

Next, we move to the detection of column types. We exclude all `None` values and examine the remaining cells for every column. We have categorized these into nine types (ordered by priority during classification):

- **empty**: an empty column.
- **date**: various types of dates. Date parsing is based on the `dateparser` package. However, it generates a lot of false positives on our data, so we have imposed several restrictions:
  - numbers must be present;
  - if a cell is parsed as a date, either the month or a full year must be present, or the column header must contain the word *date*.

  If the whole cell is not parsed as a date but a word for month is present, we look for date substrings in the string and classify the column as `date` if any found.
- **year**: years. We use a regular expression to check if each cell contains the same number of numbers (i.e. is somehow structured) and one of them is a year. Afterwards, we take a year if it is the only number in the string. We limit years to numbers between 1700 and 2099 to avoid false positives.
- **rank**: rankings or any sequential row enumeration. The column is classified as a ranking if all cell values are integers and the difference between any two consecutive cells is 0 or 1.
- **num**: numbers (floats and integers, including negative values, thousands with delimiters, e.g., `200,000`, and percents). We use a regular expression for the detection, which matches the string containing only the number.
- **time**: durations (e.g., `33:42`). Times are parsed if all cells contain `:` and time can be parsed by the `pytimeparse` library. The processed value is the number of seconds.
- **same_n_nums**: mixed values where several numbers are present or the cells contain both numbers and alphabetic characters, but the number of numbers is the same in every cell.
- **str**: non-numeric values only, checked with a regular expression.

- **mixed**: both numeric and alphabetic characters are encountered, and no structure can be determined.

In the example provided in Figure 4.2, nearly all column types are represented, namely: *date*: `date`, *opponent*: `mixed`, *location*: `str`, *result*: `same_n_nums`, *attendance*: `num`. Another example is provided in Figure A.4, which features column types `rank`, `year`, `time`. Figure A.3 presents a column of the type `empty`.

To ensure the accuracy and sensibility of the parsing, we manually evaluated a set of 45 tables, i.e., 5 tables for each column type, and each column in every table was evaluated. In all the tables, we encountered only one arguable column type, where an increasing score was parsed as a rank. All other columns were parsed correctly in accordance with the proposed logic.

Table 6.1 shows the distributions of column types in the complete Logic2Text dataset, which was used for development, as well as the test partition of LogicNLG, which was utilized for the final evaluation. Types `str` and `num`, which are the most straightforward for further processing both technically and conceptually, comprise over a half of all columns for both sets of tables.

column type	**Logic2Text (all)**	**LogicNLG (test)**
str	0.368	0.366
num	0.225	0.217
mixed	0.134	0.143
same_n_nums	0.123	0.13
date	0.059	0.066
rank	0.043	0.041
year	0.041	0.028
time	0.005	0.007
empty	0.002	0.002

Table 6.1: Column type distribution in all Logic2Text tables and LogicNLG test tables.

6.3 Content Selection

The content selection step generates logical form templates based on the aggregated information about the table, which can potentially include the header, column types, and column statistics. This step aims at solving the problem of excessive input (by avoiding the need to pass the entire table to the model) and plays an essential role in controllable generation by predicting relevant columns and operations.

The system receives a table as input, aggregates it to the required format, and feeds it into the sequence-to-sequence model. The output is a logical form template, which is an intermediate representation formalized in LF syntax but without the values from the table. To obtain LF templates for training, we substitute all values (everything that is not included in the header or is not equal to the special token `all_words`) with a placeholder `X`. For instance, given a table as depicted in Figure 4.2, the LF template options are as follows:

- `most_eq { all_rows ; result ; X }`
- `eq { hop { argmax { all_rows ; attendance } ; location } ; X }`
- `round_eq { avg { all_rows ; attendance } ; X }`

In this section, we experiment with varying types of input information (Section 6.3.2), training data (Section 6.3.3), and generation strategies (Section 6.3.4). We do experiments incrementally, measuring the performance of the models with several validity and variability metrics and reporting the results in each experiment. Evaluation metrics are described before all experiments in Section 6.3.1. The final results are analyzed in Section 6.3.5.

6.3.1 Evaluation

Generating templates is quite non-trivial task. Unlike conventional sequence-to-sequence tasks, where the primary metric group measures the match with a reference, our goal is different. The model should not be designed to produce outputs that match the reference as closely as possible. Instead, it should provide a variable yet valid set of options based on the input table aggregation. Consequently, although we report BLEU-3 and SacreBLEU scores, we have devised our own metrics that we use throughout our experiments.

The first group of metrics is related to the validity of the generated templates. These measurements are conducted by the template filling algorithm (Section 6.4). Template filling is also prone to errors and is generally stricter than the original LFs. However, we argue that the metrics below align well with the general objective since they evaluate whether the template can be used in this specific pipeline. They evaluate general syntax and overall template rationality (e.g. the template filling fails if the `only` function is applied to a column where all values are repeated).

- **syntax check (syntax)**: the proportion of generated templates that are syntactically correct out of all predictions. The metric checks the match of brackets, the validity of function names used, the correctness of the mentioned column headers and their types, and verifies the number of function arguments and placeholders.

- **executability (exec)**: the proportion of generated templates that can be successfully executed relative to all predictions. This metric measures if the templates can actually be filled. Given that it also applied to the templates with syntactic errors, the executability score is always lower than the syntax score.

As our aim is to generate multiple, maximally diverse options per input, we have designed two metrics to evaluate variability. It's crucial to note, however, that these metrics are affected by the number of generated outputs, making them unsuitable for comparisons involving different output numbers. Thus, we maintain a constant number of generated options across all our experiments.

- **table variability (var)**: the average of how many unique LF templates are generated per input table. The ratios are normalized to lie between 0 and 1, since this ratio cannot be zero originally, e.g., it would be 1/5 if all 5 inputs are the same.
- **number of generated structures (structs)**: number of unique structures (a term introduced in Section 4.2) among the executable generated LF templates.

Since we target both validity and variability, our main metric, which we maximize in our experiments, is the **average between executability and the number of unique structures (ex+st)**. The executability metric has a range from 0 to 100, and the number of unique structures is expected to be less than 150. Therefore, both metrics align in magnitude and can be averaged. However, this averaging exhibits a bias towards the number of unique structures due to its larger variation (for example, a difference between 36 and 40 structures is more prominent than a change from 77.1% to 77.6% in executability). However, we argue that this bias is in line with our objective because small fluctuations in executability can be neglected in favor of more diverse outputs.

6.3.2 Input Format

A table can be aggregated with varying levels of detail, and we explore multiple input choices that differ in the volume of information passed to the model. For all options, we include the table title, its header and the detected column types. Below, we list all input configurations we consider, providing the example of each.

- **Types only (T)**: Essential information only: the title and the table header with the associated column types. This level of detail should suffice to distinguish between numerical and string columns and possibly capture a column meaning from the column name.

  ```
  Title: wake forest demon deacons football , 1980 -
  89. Column date. Type: date. Column opponent. Type:
  mixed. Column location. Type: str. Column result. Type:
  same_n_nums. Column attendance. Type: num.
  ```

- **Basic statistics (TB)**: T + basic column statistics expressed in a generalized form without specific numbers. This input option additionally describes value distribution in the column using natural language. The aim of this approach is to capture the information required for functions that operate on precise table values, like all `*_eq` functions.

  ```
  Title: wake forest demon deacons football , 1980 - 89.
  Column date. Type: date. Stats: Column has all unique
  values, all values have one occurrence. ... Column
  location. Type: str. Stats: Column has most unique
  ```

```
values, one value has most occurrences, most values have
one occurrence. ...
```

- **Value (TBV)**: T + B + value example. The objective of this addition is to show the column format to the model, for instance, a measurement that can occur beside the numerical value.

  ```
  Title: wake forest demon deacons football , 1980 - 89.
  Column date. Type: date. Stats: Column has all unique
  values, all values have one occurrence. Value: 09 / 12
  / 1987. ... Column location. Type: str. Stats: Column
  has most unique values, one value has most occurrences,
  most values have one occurrence. Value: groves stadium
  winston - salem , nc. ...
  ```

- **Numerical statistics (TBN)**: T + B + numerical statistics where applicable. A set of aggregations depends on the column type. `max`, `min`, `avg`, `sum`, `maximum diff` are calculated for `num` type and for each number in `same_n_nums`. For `date`, `year`, and `time`, only `max`, `min`, and `maximum diff` are calculated.

  ```
  Title: wake forest demon deacons football , 1980 -
  89. Column date. Type: date. Stats: Column has all
  unique values, all values have one occurrence, first:
  1987-09-12, last: 1987-11-21, max diff days: 70, max diff
  months: 2, max diff years: 0. ... Column location. Type:
  str. Stats: Column has most unique values, one value has
  most occurrences, most values have one occurrence. ...
  ```

- **Basic, numerical statistics, and value (TBNV)**: All input options combined.

  ```
  Title: wake forest demon deacons football , 1980 -
  89. Column date. Type: date. Stats: Column has all
  unique values, all values have one occurrence, first:
  1987-09-12, last: 1987-11-21, max diff days: 70, max
  diff months: 2, max diff years: 0. Value: 09 / 12 /
  1987. ... Column location. Type: str. Stats: Column
  has most unique values, one value has most occurrences,
  most values have one occurrence. Value: groves stadium
  winston - salem , nc. ...
  ```

More detailed input options can be quite long which results in truncation of some inputs. However, while truncation obviously means the loss of information, it is worth noting that it is the column details that get truncated. This is less concerning compared to truncating actual table cells from the table's end, as observed in the baseline. Furthermore, it is essential to highlight that content selection operates on the whole table rather than a predefined column selection.

Table 6.2 shows the model performance on all input formats.

Format	exec	syntax	var	structs	ex+st	SacreBLEU	BLEU-3
T	68.5	88.2	**85.7**	44	56.2	59.2	54.4
TB	78.6	88.7	84.7	36	57.3	**59.4**	**55.9**
TBV	77.6	88.6	84.4	41	59.3	58.2	55.0
TBN	**78.7**	**88.9**	84.6	44	61.4	57.8	55.1
TBNV	78.1	87.6	84.2	**45**	**61.5**	59.1	55.7

Table 6.2: Performance of models trained on different input formats.

The results indicate that there are no huge differences between different input formats, although T option (only headers and column types) displays notably lower executability. When considering the more detailed input options, we find that $\approx 11\%$ of outputs are syntactically invalid, and additional $\approx 10\%$ cannot be filled with the values, as the executability score shows. Overall, $\approx 78\%$ of the outputs are executable, suggesting that, on average, 4/5 outputs per table can be filled with the table data.

Another observation concerns variability and none of the models shows good performance in this respect. Given > 150 unique structures in the training data, the models produce ≤ 45 executable structures, which is quite low. In our subsequent experiments, we investigate whether it is possible to increase the variability while keeping executability on a satisfactory level.

In spite of the inconsistent maximum metric values observed across the models, we primarily focus on optimizing the `ex+st` metric, as stated in Section 6.3.1. While the TBN input type demonstrates superior executability, the TBNV option appears more promising in terms of variability, and we use it for further experiments.

6.3.3 Training Data

Liu et al. (2022a) provided a synthetic dataset which they employed for the table-to-LF pre-training. We use $> 675,000$ LFs from this dataset for our experiments on data augmentation, pre-training, and data balancing.

Data augmentation (all): We add all generated data to the training set and conduct basic fine-tuning without distinguishing between the original and the synthetic data.

Pre-training (pretrain): We pre-train on the synthetic dataset and further fine-tune on the original data samples. Parameters for the pre-training are listed in Appendix A.4.2.

Balancing (balance + power): We add samples from the synthetic dataset trying to mitigate the data imbalance described in Section 4.2. We use "soft" balancing, ensuring that the dominant structures still prevail and and the rare ones remain infrequent, but smoothing the imbalance. The process for the soft balancing consists of the following steps:

1. derive LF structures and obtain their counts in the dataset;

2. get a smoother distribution by raising the number to the power less than 1. The exact power is a hyperparameter and a smoother distribution is achieved by a lower power value. We further normalize the obtained numbers to lie between 0 and 1;
3. calculate new expected counts from the obtained normalized "probabilities". We do not add new samples to the most frequent structure, so we can derive an expected new dataset size from its original count and a new "probability": `int(count / new_prob)`. From the new total number and "probabilities", it is possible to get the new counts of all other structures.
4. We select structures to be added based on their original and expected counts, taking new samples from concatenated original and synthetic data and possibly repeating instances.

We try two powers: 0.5 (square root) and 0.75.

The performance of the model on different training data is presented in Table 6.3.

Strategy	**exec**	**syntax**	**var**	**structs**	**ex+st**	**SacreBLEU**	**BLEU-3**
orig	78.1	87.6	84.2	45	61.5	**59.1**	55.7
all	73.5	78.5	**97.7**	29	51.3	53.1	53.2
balance 0.75	70.6	82.0	90.0	47	58.8	54.1	52.1
balance 0.5	64.7	74.9	96.1	55	59.8	47.4	47.9
pre-train	**81.9**	**91.4**	80.1	**54**	**67.9**	58.8	**55.9**

Table 6.3: Results of models trained on different training data and using different training strategies. *orig* refers to the best model from section 6.3.2, which is trained only on the original data.

The results indicate that training on all data at once greatly reduces the number of unique generated structures. While the balancing stratey enhances the diversity of executable structures, there's a marked decrease in executability. This might be due to the generation of more complex or challenging templates, which our template filling algorithm finds difficult to address given our restrictions on values and data types.

In contrast, pre-training substantially enhances generation quality. This can be explained bythe model's early exposure to valid and varied data over several steps, followed by fine-tuning using natural data that features more sensible selection of columns and operations on them. For subsequent experiments, we use this pre-trained + fine-tuned model.

6.3.4 Generation Strategies

In previous sections, we adopted top-k sampling with $k = 50$ both for choosing the best checkpoint during training and for final predictions, since this strategy has shown the best executability vs. variablility balance in preliminary experiments.

We use the best model based on the experiments in the previous sections and examine several generation parameters that can potentially increase the diversity:

- varying k values in top-k sampling: only the top k values are selected from the token conditional probability distribution, and the next token is picked randomly according to the distribution. Random selection substantially increases variability but avoids generating unexpected tokens due to restricting the token set to the most probable options only. We experiment with k of 5 and 10.

- top-p sampling, chooses the next token from the set of tokens whose cumulative probability exceeds the p threshold. This type of sampling preserves diversity but accounts for the problem of sharp and smooth conditional distributions, adjusting the number of tokens taken into consideration at every step. We consider $p = 0.95$.

- temperature: a parameter for the softmax function which influences the sharpness of the resulting probability distribution. When combined with sampling techniques, a temperature below 1 sharpens the distribution, leading to a more restricted generation. Conversely, higher values tend to make the distribution smoother, enhancing diversity but possibly compromising fluency. We try a temperature of 1.5 in combination with top-k sampling with $k = 50$.

- beam search, which keeps the *n_beams* most probable hypotheses at each time step, maintaining a pruned tree of hypotheses and selecting the most probable ones at the end of the generation. This method can potentially generate high-probability sequences that have a rare token at the beginning. We set *n_beams* to 5.

- diverse beam search, which divides beams into distinct groups, enforcing diversity among beam groups throughout the generation process. We use 25 beams divided into 5 groups and set diversity penalty to 1.

The results are presented in Table 6.4.

strategy	**exec**	**syntax**	**var**	**structs**	**ex+st**	**sacreBLEU**	**BLEU-3**
k=50	81.9	91.4	80.1	**54**	67.9	58.8	55.9
temp=1.5, k=50	73.6	84.4	90.3	48	60.8	55.3	53.1
b=25, bg=5	79.4	84.7	99.7	51	65.2	**68.7**	59.0
b=5	**87.8**	**92.4**	**100**	46	66.9	61.2	**61.0**
k=5	82.0	91.4	79.2	53	67.5	58.9	56.0
k=10	81.9	91.4	80.1	**54**	68	58.8	55.9
p=0.95	82.6	91.8	78.0	**54**	**68.3**	59.2	56.1

Table 6.4: Results of different generation strategies. k stands for k in top-k sampling, p is the parameter for top-p sampling, *temp* is temperature, b is the number of beams, and *bg* is the number of beam groups.

The outcomes from our experiments are as follows:

- top-k sampling with temperature noticeably decreases executability and, consequently, the number of executable structures;
- diverse beam search produces results with satisfactory executability and almost perfect table variability, which is the main feature of the approach in general. However, a slight drop in the number of unique structures makes this method less desirable for our purposes.
- traditional beam search achieves the best executability results among all tested generation strategies. It also shows perfect table variability, which is inherent to its nature of picking top probabilities (resulting in mostly valid sequences) and keeping only distinct sequences in the tree. The downside is its limited structure diversity.
- For top-k sampling, $k = 5$ performs worse than the baseline of 50 but $k = 10$ slightly improves the result[1].
- Top-p sampling turns out to be the best generation strategy, marginally outperforming top-k sampling in terms of executability while maintaining the same count of unique structures. Nevertheless, it is important to note that table variability is slightly worse for this strategy.

6.3.5 Results

In this section, we provide the scores of several models on the test set and conduct more in-depth analysis. Table 6.5 presents the performance of several models on the test set. We show the best models from Sections 6.3.2, 6.3.3, and 6.3.4.

Parameters	**exec**	**syntax**	**var**	**structs**	**ex+st**	**SacreBLEU**	**BLEU-3**
TBNV, orig, k=50	78.7	87.9	**84.4**	51	64.9	**59.4**	55.3
TBNV, pre-train, k=50	81.1	90.5	79.3	**60**	**70.6**	58.6	55.7
TBNV, pre-train, p=0.95	**81.9**	**91.1**	77.0	57	69.5	59.1	**56.0**

Table 6.5: Results of the best parameter sets on the test set.

The data presented in the table shows that pre-training considerably enhances the quality of generation, but the choice of an alternative generation strategy has a negative impact on variability. The decline in `ex+st` metric on the test set (-1.1) is larger than its increase for the development set ($+0.3$). Therefore, our final choice of the content selection model is the model which takes the input with all features, is additionally pre-trained on artificial data and employs generation with top-k sampling with $k = 50$ (labelled as TBNV, pre-train, k=50).

Diving deeper into the performance of the best model on the test set, we examine its executability and variability.

[1] `ex+st` for is correctly larger and its components `exec` and `syntax` are reported to be the same due to rounding.

Syntax errors comprise around a half of all executability errors, and our analysis showed that $\approx 15\%$ of syntactic errors are caused by hallucination (e.g., non-matching column header names). The remaining 85% are caused by the data type mismatch between the column and the selected function. It is worth noting that there are no errors due to brackets mismatch, indicating model's proficiency in generating the LF structure. A syntactically correct template may fail to be filled because of the absence of values or other conditions required by the function or due to our template filling restrictions imposed to keep the sensibility of the resulting logical forms (Section 6.4).

In general, executability of 81.9 means that, on average, 4 out of 5 templates can be filled, but the actual number of executable templates varies for different tables. Below, we provide an example of generated options for a table, two of which are valid and three are not:

---

test 416, `2-11895475-1`, Figure A.5

Executable: a statement that some *venue* occurs only once on some date (several options available):

```
and { only { filter_eq { all_rows ; venue ; X } } ; eq { hop { filter_eq {
all_rows ; venue ; X } ; date } ; X } }
```

Executable: a statement that the same competition was held on two different dates:

```
and { eq { hop { filter_eq { all_rows ; date ; X } ; competition } ; hop {
filter_eq { all_rows ; date ; X } ; competition } } ; and { eq { hop {
filter_eq { all_rows ; date ; X } ; competition } ; X } ; eq { hop {
filter_eq { all_rows ; date ; X } ; competition } ; X } } }
```

No options – a statement that most competitions are X, but there are no prevailing values in the *competition* column, so `most_eq` function cannot be filled and executed:

```
most_eq { all_rows ; competition ; X }
```

No options – a statement that some competition happened only once on some date, but all competitions occur at least twice, so `only` function fails:

```
and { only { filter_eq { all_rows ; competition ; X } } ; eq { hop {
filter_eq { all_rows ; competition ; X } ; date } ; X } }
```

Syntax – a statement about maximum score, but *score* has the column type of `same_n_nums`, which is incompatible with `max` operation:

```
eq { max { all_rows ; score } ; X }
```

Table variability of 0.77 implies that there are 4 unique generations our of 5 in total per table, and the resulting number of unique executable structures is 60, which is acceptable. However, these metrics do not measure imbalance in the selection of functions and structures, which we discussed regarding the training data (Section 4.2). Therefore, we compare the function and structure distributions on the test set and our generated outputs. We take only executable templates from the prediction pool, and the size of the resulting prediction set is 2028 (out of initial 2500 for 500 tables). As stated in Section 4, the size of the test set is 1092.

Table 6.6 compares top occurring structures in the original test data and our prediction set. It can be clearly seen that the model has a very strong preference towards generating the structure with `only` function, producing it in over 35% of cases. Apart from generation process, this prevalence can be explained by the ease of template filling, since unique values are quite common in columns and there are no additional restrictions for this structure. Most of other structures have proportions similar to the test data, except for `less` function which is in top-5 of generated structures, and `argmax` which occurs often in the test data but not in the model outputs.

structure	**data**	**preds**
`and { only { filter_eq { all_rows ; Y ; X } } ; eq { hop { filter_eq { all_rows ; Y ; X } ; Y } ; X } }`	0.102	0.362
`most_eq { all_rows ; Y ; X }`	0.097	0.090
`eq { count { filter_eq { all_rows ; Y ; X } } ; X }`	0.128	0.075
`less { hop { filter_eq { all_rows ; Y ; X } ; Y } ; hop { filter_eq { all_rows ; Y ; X } ; Y } }`	0.044	0.070
`round_eq { avg { all_rows ; Y } ; X }`	0.076	0.064
`eq { hop { argmax { all_rows ; Y } ; Y } ; X }`	0.093	0.016

Table 6.6: Proportions of top-occurring structures in the original test data (*data*) and predictions (*preds*).

We also investigate the frequencies of individual functions. Figure 6.2 depicts the function distribution in the original test data and generated LF templates. The comparison highlights a bias towards `only` operation as well, which is also presumably a reason for higher `and` and `filter_eq` frequencies as well. Other findings include wider occurrence of several functions (`sum`, `most_greater`) and unexpected rarity or absence of others: `count` is generated less frequently than it is used in the data, `argmax` and `greater` are very infrequent, and `nth_argmax` and `diff` are never generated although present in the data.

Theoretically, several underrepresented functions could be added to the outputs by rule-based output augmentation. For instance, all templates with `less` function can be converted to the analogous templates with `greater` or used to construct a template with `diff` operation. An alternative could be more targeted balancing in the training data. However, these steps remain out of scope of this work.

6.4 LF Template Filling

This section describes the template filling algorithm, which generates valid logical forms (LFs) from the templates created in the previous stage. An example illus-

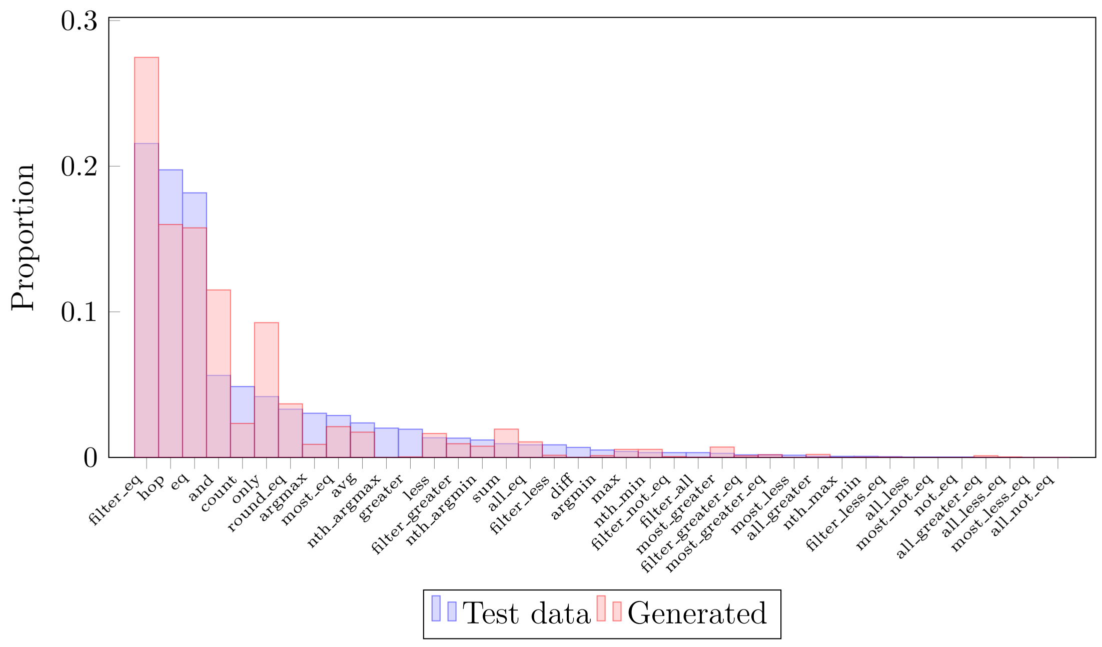


Figure 6.2: Function distribution comparison between Logic2Text test set and generated outputs.

trating the inputs and outputs is provided below. The input suggests selecting `n` for `nth_argmax` and filling the corresponding value from `home team` column. The reference takes the second-largest `crowd` value, while our system suggests three alternatives using the first, second, and third-largest values.

---

train 79, `2-10750694-12`[2]

Input:

```
eq { hop { nth_argmax { all_rows ; crowd ; X } ; home team } ; X }
```

Suggested options:

```
eq { hop { nth_argmax { all_rows ; crowd ; 1 } ; home team } ; geelong }
eq { hop { nth_argmax { all_rows ; crowd ; 2 } ; home team } ; south melbourne }
eq { hop { nth_argmax { all_rows ; crowd ; 3 } ; home team } ; carlton }
```

Reference:

```
eq { hop { nth_argmax { all_rows ; crowd ; 2 } ; home team } ; south melbourne }
```

in the 1952 vfl season , the 2nd largest crowd happened when the home team was south melbourne .

In general, we traverse LF as a tree structure, following a depth-first search (DFS) approach, and during this traversal, we expand the tree of potential filling options (Section 6.4.1). We have implemented nearly all the functions presented in the data (Section 6.4.2). Each of these functions returns a list of possible valid filling

[2]Examples of the algorithm outputs in this section use the following structure: table identifier (split and index in the dataset, table id); input; suggested options; reference LF; reference text. General LF cases from the dataset are presented with table identifier, reference LF, and reference text.

options, along with the result of function execution for every option. If a function is an argument of another function, the parent function is executed on each of the child's options, thereby expanding the options tree further.

Moreover, certain structures in the data require consistency across different parts of the LF. Therefore, we perform additional consistency checks that rely on other LF parts and select placeholder values that satisfy a set of conditions (Section 6.4.3). Lastly, to enhance the quality and quantity of filling options, we employ additional value processing (Section 6.4.4).

The evaluation strategies are described in Section 6.4.5, and the results are discussed in Section 6.4.6.

6.4.1 Filling Algorithm

First, a recursive syntactic check is conducted. It includes:

1. technical LF parsability, which implies the presence of function names in the appropriate locations and a match between the opening and closing brackets.
2. validity of function names;
3. validity of all arguments, which includes nested structures (evaluated through a recursive call), special keywords (such as `all_rows` and placeholders), and column names.
4. applicability of the function to the type of column on which the function is operating;
5. the match between the number of provided arguments and the function definition;
6. number of placeholders, which can not exceed one.

Examples of several syntactically invalid functions are below:

- Brackets mismatch:
  `eq { hop { argmax { all_rows ; tonnage } } ; name } ; X }`
- Not suitable column type:
  `eq { hop { argmax { all_rows ; episode name } ; actor } ; X }`
- Mismatch in the number of arguments (`argmax` does not need a value to be filled):
  `eq { hop { argmax { all_rows ; tonnage ; X } ; name } ; X }`

If the check is successfully passed, the recursive algorithm starts. One recursive step is described below:

1. For a given function, we iterate over its arguments and collect all possible combinations of the arguments that can be filled at this stage. The variability comes from processing nested LFs at this step: a recursive call is

made to process, execute and fill the nested LF, returning all possible filling options. If a placeholder is present, it is left unchanged at this stage. However, each resulting option constitutes a complete set of arguments ready for the selection of the placeholder value and the execution of the current function.

2. For each obtained argument combination, we call the function and get the list of filling options along with the corresponding execution outputs. We then fill the LF template and return the function's output and a filled template for every option.

6.4.2 Functions

In this study, function implementations serve not only for execution but also for producing all valid value options for filling the function placeholder, if required.

We have implemented all functions from Table 4.3 except for `all_not_eq`, which is encountered in the dataset only once, stating that all values in the column are not `None` (train 3450, `2-17593350-2`), and has a relatively unclear semantics.

Functions are called in a uniform manner and yield a uniform output. Apart from compulsory function arguments, functions connected with other LF parts also take information about the current state of LF tree in order to perform consistency checks. Functions that do not have placeholders (e.g., `max`, `count`, and others) perform only execution on the given input, and functions that require to be filled (e.g. `nth_max`, `filter_eq`) also return filling options along with corresponding execution results.

An overview of function implementations is presented in Table 6.7. Overall, the implementations, including column type and value restrictions, align with the semantics of the functions. The *additions* column in the table also lists which consistency checks are run for every function and which helpers are used to derive value options. Non-trivial details about the implementation are marked in the table and described in Appendix A.5.

6.4.3 Consistency Checks

Several LF structures in the data require consistency across different LF parts. Consistency checks are relevant for filtering functions, which return a subtable or a row, to ensure that filtering is the same in different LF parts or that both arguments satisfy a condition for an ancestor boolean function. In this section, we will describe each check and provide relevant examples.

Uniqueness The value must occur only once in the column if ancestor functions include `only` (which by its semantics requires the value to be unique) or `hop` (which accepts a row thus implying value uniqueness in preceding filtering function). Otherwise, the value must occur more than once to make subsequent operations more sensible. In the following example, value *f1* in `filter_eq` is checked to be unique before being selected:

function	column types	returned	filled	additions
filter_all	all	original table	None	
filter_eq	all	filtered table	raw value[E1]	all checks[6.4.3], substrings[6.4.4]
filter_not_eq	all	filtered table	raw value	all checks
filter_greater_eq **filter_less_eq**	num, date, rank, year, time[E4]	filtered table	raw value	all checks
filter_greater **filter_less**	num, rank[E6]	filtered table	floor ceil	all checks, rounding[6.4.4]
max **min**	num, date, rank, year, time	raw value[E2]	None	
argmax **argmin**	num, date, rank, year, time	row	None	
nth_max[E10] **nth_min**	num, date, rank, year, time	raw value	n	
nth_argmax **nth_argmin**	num, date, rank, year, time	row	n	check for and[E11]
avg	num, time[E7]	rounded num	None	
sum	num, time[E7]	rounded num	None	
count	all	int	None	
diff	num, date, rank, year, time[E9]	rounded num or timedelta in days/seconds	None	
hop	all	raw[E2] / proc value	None	check if parent func is eq[E2]
only	all	bool[E3]	None	
and	all	bool	None	
eq	all	bool	raw value / None[E12]	
not_eq	all	bool	None	
round_eq	num, time	bool	rounded num	
greater[E9] **less**	num, date, rank, year, time	bool	None	
most_eq[E13]	all	bool	raw value	substrings
most_not_eq	all	bool	raw value	
most_greater_eq **most_less_eq**	num, date, year, time[E8]	bool	raw value	
most_greater **most_less**	num	bool	floor ceil	rounding
all_eq	all	bool	raw value	substrings
all_not_eq				
all_greater_eq **all_less_eq**	num, date, year, time[E8]	bool	raw value	
all_greater **all_less**	num	bool	floor ceil	rounding

Table 6.7: Overview of function implementations. Hyperlinks next to certain elements lead to the comments about their implementation (Appendix A.5).

train 393, 1-26996293-2

```
only { filter_eq { all_rows ; position ; fl } }
```

in the second round of the 1970 cfl draft , there was only one fl or flankerback drafted .

Greater / Less This check is conducted if the ancestors in the LF tree contain a `greater` or `less` function, where the second argument must be strictly less / greater than the first one. Here, *baker brook* is verified against *madawaska* and confirmed to have a smaller *area km 2* value:

train 412, 2-171250-2

```
greater { hop { filter_eq { all_rows ; official name ; madawaska } ; area km 2 } ;
hop { filter_eq { all_rows ; official name ; baker brook } ; area km 2 } }
```

the parish of madawaska (madawaska county , new brunswick) has a larger land area than the parish of baker brook .

Inequality This condition is required by `not_eq` and `diff` functions, and the second value must not be equal to the first argument. In the example below, *end of the middle* is selected because it has a different *original air date* from *joke overload.*

train 334, 1-28081876-4

```
eq { diff { hop { filter_eq { all_rows ; title ; joke overload } ; original air
date } ; hop { filter_eq { all_rows ; title ; end of the middle } ; original air
date } } ; -7 }
```

the childrens hospital episode titled " joke overload " had an original air date that was 7 days before the original air date for the episode titled " end of the middle . " .

Equality The equality check is run for the special case of `eq` function, where both arguments are LFs and the equality between their outputs must be verified. We additionally verify that the second LF is not a full repetition of the first one. In the following example, the check must verify that the second `filter_eq` fills in a different value of *class* than the filtering in the first argument of `eq`, but their *quantity made* values must be equal.

train 441, 2-18620528-14

```
eq { hop { filter_eq { all_rows ; class ; a - 4 } ; quantity made } ; hop {
filter_eq { all_rows ; class ; a - 3 } ; quantity made } }
```

the northern pacific railway locomotive 's class a - 4 and class a-3 had the same amount of quantity made .

And This check runs if the ancestors in the tree contain `and` function. Based on the data, it must match `filter_*` functions and `nth_arg*` with `nth_*` functions across the LF. In a simple case, there is only one `and` and the first argument contains only one filtering. There are harder cases with functions like `greater` with two different filterings or nested `and` functions. In the simple example below, filtering value *slovaks* must be repeated in both `filter_eq` function calls.

---

train 1, `1-2562572-12`

```
and { only { filter_eq { all_rows ; largest ethnic group ( 2002 ) ; slovaks } } ; eq
 { hop { filter_eq { all_rows ; largest ethnic group ( 2002 ) ; slovaks } ;
settlement } ; pivnice } }
```

pivnice is the only settlement in vojvodina with slovaks as the largest ethnic group .

6.4.4 Helpers

In order to broaden the option space and produce more human-like outputs, we have developed two helper functions that transform cell values: substring matching and a special rounding algorithm.

Substring matching Substring matching aims at matching a cell if a substring in it is a match. From the raw cell values, we extract 1-, 2-, and 3-grams that do not contain punctuation and stopwords and are encountered in 3 or more cells. Afterwards, we remove all n-grams that occur only within larger selected substrings, including the original text. The resulting list of substrings is used along with the full values in the filling functions. In the example provided, the reference contains substring matching and the system proposes both whole values and substrings:

---

train 313, `2-15985163-2`

Input:

```
eq { count { filter_eq { all_rows ; method ; X } } ; X }
```

Suggested options:

```
eq { count { filter_eq { all_rows ; method ; choke } } ; 3 }
eq { count { filter_eq { all_rows ; method ; decision ( unanimous ) } } ; 4 }
eq { count { filter_eq { all_rows ; method ; decision } } ; 5 }
eq { count { filter_eq { all_rows ; method ; tko ( punches ) } } ; 2 }
eq { count { filter_eq { all_rows ; method ; tko ( referee stoppage ) } } ; 2 }
eq { count { filter_eq { all_rows ; method ; tko } } ; 9 }
eq { count { filter_eq { all_rows ; method ; submission } } ; 4 }
eq { count { filter_eq { all_rows ; method ; submission ( guillotine choke ) } } ;
 2 }
```

Reference:

```
eq { count { filter_eq { all_rows ; method ; submission } } ; 4 }
```

4 of jake o'brien 's fights have ended due to submission .

Rounding Reasonable value rounding is required for the natural and logical filling of `*_greater` and `*_less` functions. For instance, `74` can be rounded to `70`, and it is logical to round `143640` up to `150000`. Therefore, for each set of values, we find a suitable increment to a multiple of which values are rounded. The increment is 10 for the first example and 50000 for the second one. During value selection for a particular function, we round numbers up or down to the closest multiple of the chosen increment. In the following example, the increment was chosen to be 5000 and the values were rounded to its multiples:

---

train 690, `1-27764201-2`

Input:
```
eq { count { filter_greater { all_rows ; attendance ; X } } ; X }
```
Suggested options:
```
eq { count { filter_greater { all_rows ; attendance ; 10000 } } ; 6 }
eq { count { filter_greater { all_rows ; attendance ; 15000 } } ; 4 }
eq { count { filter_greater { all_rows ; attendance ; 25000 } } ; 3 }
```
Reference:
```
eq { count { filter_greater { all_rows ; attendance ; 10000 } } ; 6 }
```
the 2005 cologne centurions season featured an attendance of more than 10000 fans six times .

6.4.5 Evaluation

To assess the performance of our template filling algorithm, we employ an automatic metric that measures recall, i.e., how many original LFs our system can reconstruct from the corresponding templates. We evaluate recall separately on the training + development sets, which we utilized extensively for testing and debugging throughout the development process, and on the test set, which we did not use during the development stage. Additionally, we examine the first 50 erroneous examples from the beginning of the test set[3] to identify the reasons for unsuccessful template filling.

Furthermore, we conduct a small-scale manual evaluation to assess precision, i.e., the number of false positives encountered among the resulting filling options. We manually examine the first 50 examples of filled templates and the same 50 erroneous examples (as used for the recall analysis) to estimate the quantity of false positive cases.

6.4.6 Results

The results of the automatic recall evaluation are presented in Table 6.8.

From 50 erroneous examples, we observe several error patterns that prevent the

[3]We select the first examples because the data is initially shuffled.

split	correct	total refs	recall
train + dev	5810	8235	0.71
test	782	1092	0.72

Table 6.8: Recall of template filling algorithm on train + development and test sets. *correct*: number of items in the data where actual reference is contained in the set of suggested filling options.

template from being filled:

1. 20 examples – column type mismatch. In the example below, *score* column is of type `same_n_nums` with the format `0 - 1`, on which our system does not conduct numerical operations.

   ---

   test 32, `2-11048203-1`, Figure A.7

   Input:
   ```
   eq { hop { nth_argmax { all_rows ; score ; X } ; competition } ; X }
   ```
   Suggested options: none

   Reference:
   ```
   eq { hop { nth_argmax { all_rows ; score ; 2 } ; competition } ; 2007 afc
   asian cup qualification }
   ```
   of the games that hatem aqel played from 2002 to 2013 , the one with the second highest winning score was the 2007 afd asian cup qualification game on 22 february 2006 .

   ---

2. 9 examples – errors in substring selection in cell values. In the following example, the system did not include the option in the reference due to our restriction to take only ngrams that occur 3 times or more, and *dorell wright* occurs only twice.

   ---

   test 40, `2-13762472-8`, Figure A.9

   Input:
   ```
   eq { count { filter_eq { all_rows ; high points ; X } } ; X }
   ```
   Suggested options:
   ```
   eq { count { filter_eq { all_rows ; high points ; dwyane wade } } ; 4 }
   ```
   Reference:
   ```
   eq { count { filter_eq { all_rows ; high points ; dorell wright } } ; 2 }
   ```
   dorell wright had two high points performances for the miami heat .

   ---

3. 8 examples – number rounding mismatch. This is a usual number rounding which we also regulate, but it is not related to the helper function described above. We do not round the aggregation results to integers, causing errors for some examples:

test 148, `2-18974097-6`, Figure A.11

Input:

```
round_eq { avg { all_rows ; enrollment } ; X }
```

Suggested options:

```
round_eq { avg { all_rows ; enrollment } ; 737.88 }
```

Reference:

```
round_eq { avg { all_rows ; enrollment } ; 738 }
```

the average student enrollment of schools in the ohio river valley - western indiana conference is 738 .

4. 6 examples – restrictions imposed by us in our function implementation. In this example, restriction on `n` in `nth_argmin` (described in Appendix A.5) caused the reference not to be included in the option list.

test 18, `2-18662700-3`, Figure A.6

Input:

```
eq { hop { nth_argmin { all_rows ; time ; X } ; country } ; X }
```

Suggested options:

```
eq { hop { nth_argmin { all_rows ; time ; 1 } ; country } ; new zealand }
eq { hop { nth_argmin { all_rows ; time ; 2 } ; country } ; germany }
```

Reference:

```
eq { hop { nth_argmin { all_rows ; time ; 3 } ; country } ; united states }
```

the rowers from the united states had the 3rd shortest time in women 's double sculls during the 2008 summer olympics .

5. 4 examples – arguable or wrong reference. Here the reference includes *90 repeating* which is a loose interpretation of the resulting average, which is $43/11 = 3.9090...$:

test 83, `2-167482-1`, Figure A.10

Input:

```
round_eq { avg { all_rows ; mult } ; X }
```

Suggested options:

```
round_eq { avg { all_rows ; mult } ; 3.91 }
```

Reference:

```
round_eq { avg { all_rows ; mult } ; 3.90 90 repeating }
```

the average mult of all models is 3.90 with the 90 repeating .

6. 3 examples – other reasons.

Out of the 50 examples we analyzed, our algorithm suggested other valid options for 25 of them.

The manual analysis of the same 50 erroneous examples and the first 50 correctly filled examples reveals that our algorithm produced undesirable filling options, or false positives (FPs), for 7 out of 100 templates. To calculate precision, we employ the strictest evaluation criterion. For each template associated with a table, a set of filled options is provided by the algorithm. If at least one filling option is incorrect or does not make much sense, the entire set is considered a FP for that template. In this setup, precision is 93/100 = 0.93.

In this sample, FPs appear because of the imperfect substring matching algorithm, which tends to extract irrelevant information from the cells (such as first names, numbers, or generic parts in location names like *stadium*). The example below illustrates a set of filling options where one filling is correct while the other one is a FP (*park*, which is too generic for a location).

---

test 39, `2-11788447-2`, Figure A.8

Input:
`eq { count { filter_eq { all_rows ; venue ; X } } ; X }`

Suggested options:
`eq { count { filter_eq { all_rows ; venue ; park } } ; 4 }`
`eq { count { filter_eq { all_rows ; venue ; roker park } } ; 2 }`

Reference:
`eq { count { filter_eq { all_rows ; venue ; roker park } } ; 2 }`
roker park was used two times as a venue during 1990-1991 season when kieron brady played .

6.5 LF-to-Text Generation

The objective of this step is to generate a sentence from a logical form. Consider the example below:

---

train 5215, `2-1235920-4`

Title:
julian bailey

LF:
`eq { count { filter_eq { all_rows ; team ; mg sport & racing ltd } } ; 2 }`

Output:
julian bailey drove with the team mg sport & racing ltd for a total of two years .

In this stage of the pipeline, we fine-tune a pre-trained sequence-to-sequence model in a basic setup without any additional enhancements. Our hypothesis is that modern language models, even of moderate size, are generally capable of copying values from the input and learning direct natural language counterparts of formal functions.

6.5.1 Model Description

We feed the title, linearized header, and LF to the model. During training, we use a gold standard LFs provided in the dataset, and the inference mode uses the output from the template filling step (Section 6.4). Below is an example of the input:

```
Title: list of cities , towns and villages in vojvodina. Header: settlement
 | cyrillic name other names | type | population ( 2011 ) | largest ethnic
 group ( 2002 ) | dominant religion ( 2002 ). Logical form: and { only
{ filter_eq { all_rows ; largest ethnic group ( 2002 ) ; slovaks } } ;
eq { hop { filter_eq { all_rows ; largest ethnic group ( 2002 ) ; slovaks
 } ; settlement } ; pivnice } }
```

The output is a single sentence in natural language:

> *pivnice is the only settlement in vojvodina with slovaks as the largest ethnic group .*

We generate only one option per LF, prioritizing fidelity over variability at this step.

We do not carry out any experiments at this stage as our model is quite basic but still shows good performance in this setup. We utilize the `t5-base` model, and the training and generation parameters are detailed in Appendix A.4.3.

6.5.2 Evaluation

For evaluation, we employ the BLEU score in the BLEU-3 and SacreBLEU implementations, which are also used for end-to-end evaluation. We also use BLEC score (Shu et al., 2021).

BLEC (Bidirectional Logic Evaluation of Consistency) is a rule-based metric specifically designed to evaluate the logical consistency between semantic parses and the generated texts. The output is the list of mismatched items, from which binary consistency score can be derived: score is 1 if the list is empty and 0 otherwise. The rationale behind this metric aligns with our objective as it is based on the idea that many values from the LF should directly correspond to tokens in the text.

The metric has a fixed set of target tokens (for instance, *lowest* for the `min` function, or *majority* for the `most_eq` function) and a number parser. During evaluation, it matches tokens and numbers between the formal representation and text in a bidirectional manner, and if it finds at least one missing counterpart for a target token or number, the pair is marked as inconsistent.

6.5.3 Results

Quantitative results on the test set of Logic2Text are presented in Table 6.9.

BLEC	BLEU-3	SacreBLEU
90.48	36.51	36.21

Table 6.9: LF-to-Text model performance on Logic2Text test set.

Since the BLEC metric is rule-based, we hypothesize that it might have low recall, predicting 0 for the outputs that are actually correct. We have run the metric through the original test set data, i.e., taking references as generations, and got the score of 89.2. Given that all items in this set match the input LFs, thus being true positives, this score equals to the metric recall.

We have also conducted a manual analysis in order to obtain insights about our system and metric predictions. We evaluate 50 random examples for consistent and inconsistent outputs as predicted by BLEC. Out of the positive predictions, 6 examples out of 50 were found to be incorrectly flagged as correct LF-to-text generations. Out of the negative predictions, only 21 examples out of 50 were found to be truly incorrect LF-to-text generations.

Below are examples of generations with differing BLEC predictions.

- Generation is correct, BLEC correctly predicts 1:

---

test 3, `2-16660997-15`

Title:
1963 new york giants season

LF:
```
eq { count { filter_eq { all_rows ; date ; october } } ; 4 }
```

Generation:
four of the games took place in the month of october during the 1963 new york giants season .

Reference:
during the 1963 new york giants season , the new york giants played four games during the month of october .

- Generation is wrong (`diff` is misintepreted), BLEC incorrectly predicts 1:

---

test 87, `2-18880596-3`

Title:
hughes hall college boat club

LF:
```
eq { diff { hop { filter_eq { all_rows ; year ; 2009 } ; finish position } ;
hop { filter_eq { all_rows ; year ; 2008 } ; finish position } } ; -4 }
```

Generation:
the hughes hall college boat club finished in 4th place in 2009 when compared to 2008 .

Reference:
in 2009 , the hughes hall college boat club finished four positions better than in 2008 .

- Generation is wrong (`nth_argmax` on *time* column is interpreted as *fastest* instead of *slowest*), BLEC correctly predicts 0:

---

test 34, `2-18662686-4`

Title:
rowing at the 2008 summer olympics - men 's double sculls

LF:
```
eq { hop { nth_argmax { all_rows ; time ; 2 } ; country } ; slovenia }
```

Generation:
slovenia had the second fastest time in the 2008 summer olympics - men 's double sculls .

Reference:
for the teams who have times , the team with the 2nd slowest time in rowing at the 2008 summer olympics is slovenia .

- Generation is correct, BLEC incorrectly predicts 0:

---

test 123, `2-11898258-5`

Title:
list of largest airlines in south america

LF:
```
eq { hop { argmin { all_rows ; rank } ; airline } ; tam }
```

Generation:
tam is the airline that ranks the highest among the largest airlines in south america .

Reference:
of the largest airlines in south america , the one with the best rank is tam .

---

7 End-to-End Evaluation

In this section, we analyze the outputs of our system on development and test sets of LogicNLG. We collect the generation statistics, calculate automatic table- and reference-based metrics, and compare the results with the baseline. Afterwards, we conduct a small-scale human evaluation to further examine the resulting quality of our system and reliability of the metrics. We also discuss the limitations of the automatic metrics used in previous works and this study.

7.1 Setup

We generate 5 LF templates per table and provide all possible filling options for each executable template without any pruning at the generation stage. The number of final outputs for each table depends on the executability of the generated LF templates and the number of filling options for every template, and it is highly variable.

Some LF templates tend to produce a huge number of filling options. One such example is filling options for `greater` function, for which the number of all valid comparisons from the table can exceed one hundred. However, generations derived from the same template are not versatile, and this type of output is quite far from the the real-life usage. Therefore, in our analysis, we do option selection and report four types of the output sets:

- **all**: all outputs without any selection.
- **several**: we select 5 filling options per template, keeping all templates represented but balancing their occurrences.
- **single**: we keep only one filling option per template, thus making the number of generations equal to the number of unique executable templates.
- **one**: we employ *single* strategy and subsequently select one item per table.

Distinguishing between these setups allows us to evaluate our system from three perspectives: our whole output as the most formal evaluation setup, more real-life situation when a user is suggested to select one item out of several similar options, and the smallest but the most diverse output without similar options at all. The last setup, *one*, corresponds to the selection of outputs for manual evaluation.

It is important to note that in rare cases our system does not produce any outputs for a table. In such cases, we consider the prediction pool to consist of one empty prediction.

7.2 Automatic Evaluation

For LogicNLG, we do not have golden data for the intermediate steps. Therefore, we only evaluate the executability and versatility of content selection outputs and calculate end-to-end metrics, namely, TaPEx-acc, TaPas-acc, and BLEU scores.

We keep the original BLEU score setup but change TaPEx-acc and TaPas-acc calculation. We discard the standard averaging setup because, as mentioned above, our system produces a variable number of statements per table, thus causing the discrepancy in the number of outputs between our system and the baseline and making the comparison of averages unfair. Instead, we consider an average of averages, where the NLI models' scores are averaged for each table and the overall average is taken among the table scores rather than individual examples. Therefore, the comparison is valid since the second step of the calculation is conducted on the equal number of items. In the next sections, we refer to the average of averages as TaPEx-aa and TaPas-aa.

For all option selection setups except for *all*, we run the evaluation for 10 times and report the average and the standard deviation of the scores.

7.3 Manual Evaluation

Since the task is quite difficult and the automatic metrics are not perfect, we also conduct a small-scale manual evaluation. We hired 5 NLP Master's students who evaluated a baseline and our final system by two criteria:

- Logical correctness: whether the sentence is logically correct in relation to the given table and its title. This criterion concerns only formal logical correctness, almost excluding commonsense. Nevertheless, we include "partially correct" option, possible examples of which are non-critical misinterpretation of columns (e.g., *player* instead of *team*) or the incomplete information given in a sentence, which makes it misleading (e.g., listing only several items out of all that satisfy a condition). We also add the option "nonsense" for examples that are not fluent or just do not correspond to the table at all (e.g., *In 1981, the champion was a champion.*).
- Interestingness: whether the statement describes an interesting fact about the table. We consider a statement interesting if it could be encountered in a human-produced description of a table, e.g., in a report or a news article. This formulation implies that the sentence sounds natural in the context of the table and describes a non-trivial fact that a human would also find worthy to report. We make the evaluation of this criterion relative, i.e., making the annotator select more interesting option among two of

them, in order to to facilitate the task for the annotators and get the direct comparison between the baseline and our system. Nevertheless, we include the options "both are interesting" and "none is interesting" to account for cases when both generations are equally good or bad.

For evaluation, we randomly sample 100 tables from the test set of LogicNLG. For each table, we employ *one* strategy described above. We derive an example for evaluation through *single* selection strategy since it completely mitigates the imbalance caused by numerous options per template, making the selection of less populated templates more probable and thus increasing the overall sample variability.

We developed a simple web application for the questionnaire. We divided the set of examples into four parts of 25 tables each to make it more convenient for the annotators. Exact instruction for the annotators and an example of a question are presented in Appendix A.6.

7.4 Results

In this subsection, we analyze the quantity of the resulting outputs, the results reported by automatic metric, and the insights that manual evaluation provides.

7.4.1 Generation Statistics

The results on the overall number of generated statements are presented in Table 7.1.

	dev			**test**		
	total	**avg**	**empty**	**total**	**avg**	**empty**
baseline	4260	5.0	0	4305	5.0	0
all	30443	35.9	5	34215	39.7	5
several	8275	9.8		8331	9.7	
single	2826	3.3		2873	3.3	
one	843	0.99		857	0.99	

Table 7.1: Number of statements generated by the baseline and the proposed system. *total*: raw number of generated sequences. *avg*: average number of statements per table. *empty*: number of tables with no generated statements.

For the baseline, the number of statements corresponds to the number of entries in the dataset, and the number of sentences per table is ≈ 5.

Considering our system, there are not many unique executable LF templates generated for the table, which *single* selection strategy (one option per template) reflects, producing 3.3 statements per table on average. However, some templates produce quite a lot of output options, resulting in over 35 different statements per table on average if all possible outputs are included and almost 10 for *several* strategy, where option selection is applied. For 5 tables, nothing was generated, which comprises 5/848 = 0.59% for the development set and 5/862 = 0.58% for the test set.

7.4.2 Content Selection

Table 7.2 shows that there are no huge differences in the model outputs for Logic2Text and LogicNLG. This is expected because the source of two datasets is the same and the content of the tables is similar.

parameters	**exec**	**syntax**	**var**	**structs**	**ex+st**
Logic2Text, test set	81.1	90.5	79.3	60	70.6
LogicNLG, test set	80.4	90.0	80.6	61	70.7

Table 7.2: Content selection results on the test sets of Logic2Text (Section 6.3.5) and LogicNLG.

7.4.3 End-to-end Metrics

Table 7.3 displays the performance of the baseline model and our system with three option selection strategies.

	dev						**test**					
	BLEU-3	SB	TaPEx-aa	Std	TaPas-aa	Std	BLEU-3	SB	TaPEx-aa	Std	TaPas-aa	Std
baseline	**18.0**	**17.4**	62.8	-	67.4	-	**19.1**	**18.3**	62.6	-	66.9	-
all	10.6	10.7	83.7	-	81.1	-	9.0	9.3	83.9	-	**81.8**	-
several	13.0	12.1	83.8	0.16	80.1	0.14	12.8	12.0	83.9	0.14	81.5	0.18
single	13.7	12.8	**85.5**	0.57	**81.5**	0.59	13.5	12.7	**84.9**	0.33	**81.8**	0.30
one	13.5	12.9	85.1	1.08	81.2	0.81	13.8	13.0	84.7	0.71	81.3	1.09

Table 7.3: End-to-end evaluation of the baseline model and the proposed system with four option selection strategies. *SB* stands for sacreBLEU.

The first observation is that all output variants of our system show quite high fidelity, which is substantially superior to the baseline. Both BLEU scores are much lower, which is an expected outcome, since our system is trained on another dataset, uses other logical operations and overall has slightly different underlying structure. We also find that *single* shows the best TaPEx score and quite low standard deviation.

Additionally, we compare our system to the previous works, although it is important to remember that the numbers are not directly comparable, since the output format is different (discussion in Section 4.1). Nevertheless, from the baseline comparison (Section 5) we obtained an estimate of expected difference in TaPEx score, which is 3.5 points, favouring the TabFact output format.

Table 7.4 shows the TaPEx-aa scores for several previous systems and our system with *single* option selection strategy. If we consider the difference in output format, our system still outperforms the best previous one by several accuracy points.

	Output	**BLEU-3**	**TaPEx-aa**
LoFT (Zhao et al., 2023b)	original	14.9	63.5
PLOG, T5-base (Liu et al., 2022a)	original	18.9	61.6
PLOG, BART-large (Liu et al., 2022a)	original	**21.0**	73.6
ReasTAP (Zhao et al., 2022)	original	20.7	79.4
Ours, single	tabfact	13.5	**84.9**

Table 7.4: BLEU-3 and TaPEx-aa scores for previous systems and ours with *single* option selection strategy. Please note that direct comparison with previous works is not possible due to the output format mismatch.

7.5 Manual Evaluation Results

We obtained the results on 100 tables from five annotators, and the following aspects were annotated:

1. **clarity of the given table**: binary, and no other questions were answered in case the table is unclear for the annotator;
2. **correctness**: four classes: nonsense, incorrect, partially correct, correct.
3. **interestingness**: four relative classes: none is interesting, first is more interesting, second is more interesting, both are interesting.

Five annotators marked 0, 1, 3, 4, and 16 tables as unclear, the median being 3 and the average 4.8. There is no consistency in tables marked as unclear, which can be due to different background of the annotators.

7.5.1 Logical Correctness

First, we evaluate inter-annotator agreement by measuring Fleiss' kappa. We evaluate several variations differing in granularity:

1. all tables, including the unclear ones, which results in 100 tables, and 5 categories, including *unclear* category. Agreement is 47.1.
2. tables that are clear to everyone: 80 tables, only 4 correctness categories. Agreement is 53.9.
3. tables that are clear to everyone and binary correctness categorization (nonsense and incorrect vs. partially correct and correct): 80 tables and 2 categories. Agreement is 56.4.

Overall, we can observe moderate annotator agreement, which is an acceptable level given the complexity of the task. We provide the individual annotator statistics in Table A.2 in Appendix A.7. For aggregation, we excluded unclear tables for each annotator separately and calculated the proportions of each class. We then averaged the resulting numbers for each class across all annotators. Figure 7.1 presents the comparison of the baseline and our system for every correctness class.

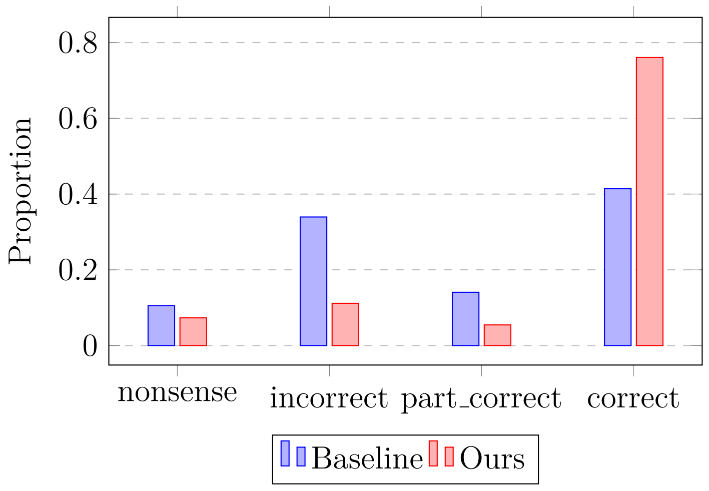


Figure 7.1: Correctness of the baseline and our system based on the manual evaluation of 100 tables.

We also estimate the correlation of the manual evaluation and TAPEX outputs. We take only clear tables and calculate the average judgement for each sample, converting *nonsense* to 0 and *partially correct* to 0.5. We further convert the result into a binary format with values ≥ 0.5 being rounded to 1.

If we consider all annotators, thus keeping 80 tables that are clear to everyone, Fisher exact test does not show significant correlation between manual evaluation and the metric output (p-value = 0.063). If we remove one annotator who discarded 16 tables, we conduct a test on 93 tables, average across 4 annotators and obtain a statistically significant correlation (p-value = 0.033). However, if evaluated individually, only one annotator has a significant correlation with TAPEX metric. We report corresponding p-values in Table A.2 in Appendix A.7.

Overall, given quite high p-values even in the cases of statistical significance, we argue that the metric does not correlate with human judgements well, even though our system shows good fidelity results in both evaluation setups. Several examples of erroneous TAPEX predictions are presented in Appendix A.8.

7.5.2 Interestingness

Interestingness annotation is much less consistent, highlighting the subjectivity of this criterion. Fleiss' kappa, if all unclear tables are excluded, equals to 15.4, which is a slight agreement. Therefore, we provide the option proportions for all annotators in Table 7.5 to show the choice pattern differences, as well as the average.

The table shows that the annotators select *none* option only occasionally. Two annotators prefer the *baseline* option, whereas other two opt for *both* and the fifth annotator selected our system most often. Overall, we can see that the

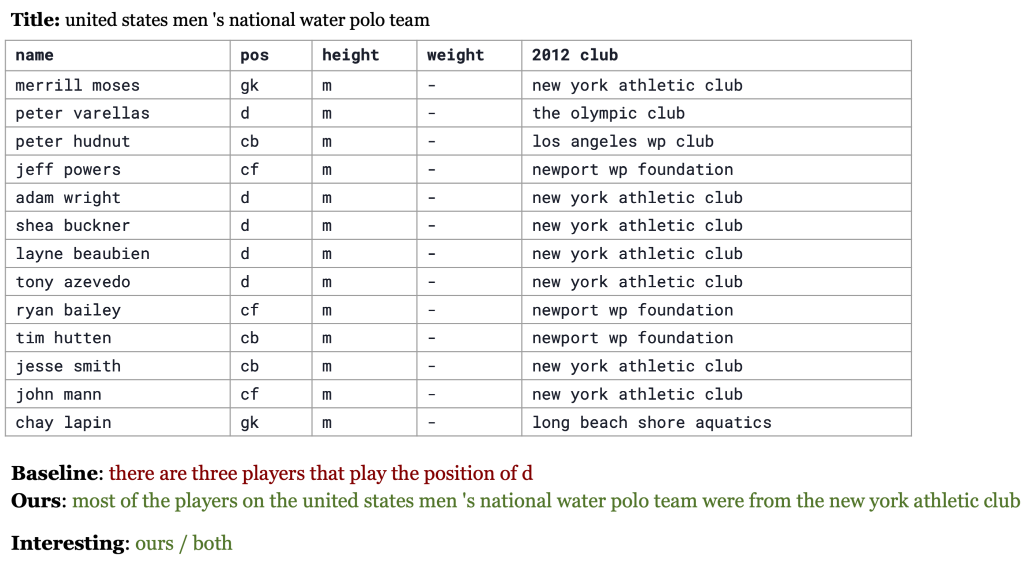
More interesting	Ann 1	Ann 2	Ann 3	Ann 4	Ann 5	avg
none	19.2	9.4	13.1	17.5	20.0	15.8
baseline	32.3	20.8	**47.6**	**29.9**	27.0	**31.5**
ours	10.1	19.8	35.7	27.8	**36.0**	25.9
both	**38.4**	**50.0**	3.6	24.7	17.0	26.7

Table 7.5: Results of manual evaluation of statement interestingness. Five annotators are reported separately, with the last column containing the average.

baseline model generates quite interesting statements and overall is considered to be natural and relevant more often. This is an expected result, since the baseline model grasps the style and general patterns of LogicNLG dataset, which are richer and more complex than the ones in our training data. Nevertheless, the sentences from the proposed system were considered interesting in more than a half of cases (*e2e* and *both* options combined), thus showing that our system is capable of generating insights and non-trivial statements.

7.5.3 Examples

Below, we show several examples from the manual evaluation set. In the first example, our system produces a faithful and an interesting statement whereas the baseline does not (LogicNLG, test 2456, id `2-18961052-1`):



Title: united states men 's national water polo team

name	pos	height	weight	2012 club
merrill moses	gk	m	-	new york athletic club
peter varellas	d	m	-	the olympic club
peter hudnut	cb	m	-	los angeles wp club
jeff powers	cf	m	-	newport wp foundation
adam wright	d	m	-	new york athletic club
shea buckner	d	m	-	new york athletic club
layne beaubien	d	m	-	new york athletic club
tony azevedo	d	m	-	new york athletic club
ryan bailey	cf	m	-	newport wp foundation
tim hutten	cb	m	-	newport wp foundation
jesse smith	cb	m	-	new york athletic club
john mann	cf	m	-	new york athletic club
chay lapin	gk	m	-	long beach shore aquatics

Baseline: there are three players that play the position of d
Ours: most of the players on the united states men 's national water polo team were from the new york athletic club

Interesting: ours / both

For the following table, our system produced a correct output but an incorrect baseline was annotated as more interesting by several annotators (LogicNLG, test 1285, id `2-14971788-1`):

Title: 1967 baltimore colts season

week	date	opponent	result	record	game site	attendance
1	september 17 , 1967	atlanta falcons	w 38 - 31	1 - 0	memorial stadium	56715
2	september 24 , 1967	philadelphia eagles	w 38 - 6	2 - 0	franklin field	60755
3	october 1 , 1967	san francisco 49ers	w 41 - 7	3 - 0	memorial stadium	60238
4	october 8 , 1967	chicago bears	w 24 - 3	4 - 0	wrigley field	47190
5	october 15 , 1967	los angeles rams	t 24 - 24	4 - 0 - 1	memorial stadium	60238
6	october 22 , 1967	minnesota vikings	t 20 - 20	4 - 0 - 2	metropolitan stadium	47693
7	october 29 , 1967	washington redskins	w 17 - 13	5 - 0 - 2	rfk stadium	50574
8	november 5 , 1967	green bay packers	w 13 - 10	6 - 0 - 2	memorial stadium	60238
9	november 12 , 1967	atlanta falcons	w 49 - 7	7 - 0 - 2	atlanta stadium	58850
10	november 19 , 1967	detroit lions	w 41 - 7	8 - 0 - 2	memorial stadium	60238
11	november 26 , 1967	san francisco 49ers	w 26 - 9	9 - 0 - 2	kezar stadium	44815
12	december 3 , 1967	dallas cowboys	w 23 - 17	10 - 0 - 2	memorial stadium	60238
13	december 10 , 1967	new orleans saints	w 30 - 10	11 - 0 - 2	memorial stadium	60238
14	december 17 , 1967	los angeles rams	l 10 - 34	11 - 1 - 2	los angeles memorial coliseum	77277

Baseline: the most points baltimore scored in a game was 38
Ours: the 1967 baltimore colts had a game against the detroit lions earlier than the dallas cowboys

Interesting: baseline / both

Below we provide one of several examples where our system produced an incorrect output but was considered more interesting by some annotators. The reason for correctness error is a misinterpreted `count` operation on the *nation* column (LogicNLG, test 1364, id `2-16142610-11`).

Title: list of 1984 winter olympics medal winners

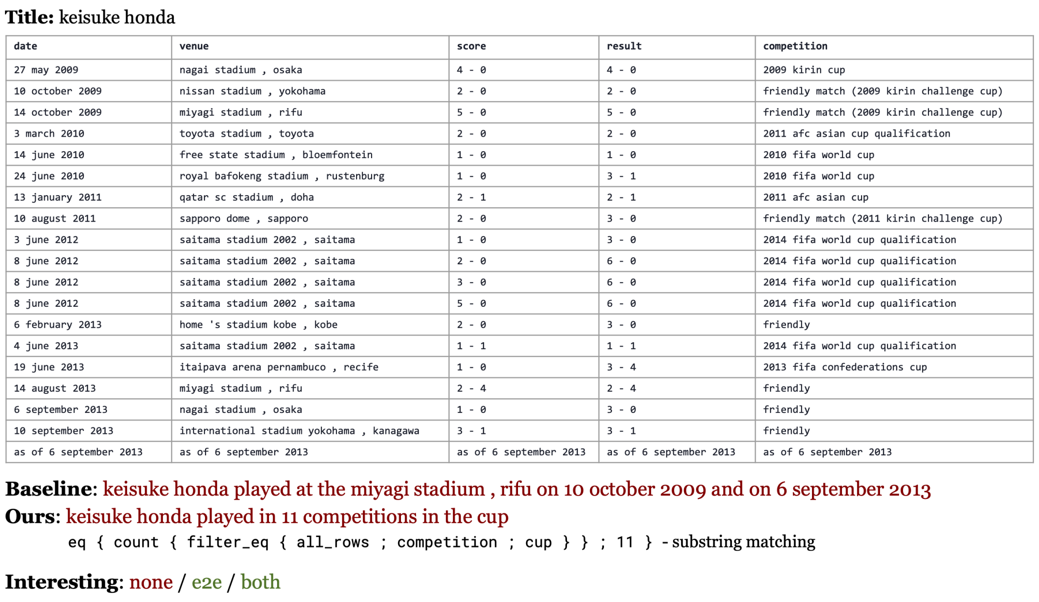
nation	sport	gold	silver	bronze	total
finland (fin)	cross - country skiing	3	0	1	4
east germany (gdr)	speed skating	2	2	0	4
sweden (swe)	cross - country skiing	2	1	1	4
canada (can)	speed skating	2	0	1	3
east germany (gdr)	speed skating	1	2	0	3
west germany (frg)	biathlon	1	1	1	3
norway (nor)	biathlon	1	1	1	3
finland (fin)	cross - country skiing	0	1	2	3
east germany (gdr)	bobsleigh	2	0	0	2
sweden (swe)	cross - country skiing	2	0	0	2

Baseline: finland won more gold medals in cross - country skiing than any other sport
Ours: east germany won three medals at the 1984 winter olympics

```
eq { count { filter_eq { all_rows ; nation ; east germany (gdr) } } ; 3 }
```

Interesting: baseline / e2e / both

Another example, where both systems fail, is the table below. Our system produces a weird output because of a false positive substring matching, discussed in Section 6.4.6 (LogicNLG, test 1227, id `2-14670286-4`):



Title: keisuke honda

date	venue	score	result	competition
27 may 2009	nagai stadium , osaka	4 - 0	4 - 0	2009 kirin cup
10 october 2009	nissan stadium , yokohama	2 - 0	2 - 0	friendly match (2009 kirin challenge cup)
14 october 2009	miyagi stadium , rifu	5 - 0	5 - 0	friendly match (2009 kirin challenge cup)
3 march 2010	toyota stadium , toyota	2 - 0	2 - 0	2011 afc asian cup qualification
14 june 2010	free state stadium , bloemfontein	1 - 0	1 - 0	2010 fifa world cup
24 june 2010	royal bafokeng stadium , rustenburg	1 - 0	3 - 1	2010 fifa world cup
13 january 2011	qatar sc stadium , doha	2 - 1	2 - 1	2011 afc asian cup
10 august 2011	sapporo dome , sapporo	2 - 0	3 - 0	friendly match (2011 kirin challenge cup)
3 june 2012	saitama stadium 2002 , saitama	1 - 0	3 - 0	2014 fifa world cup qualification
8 june 2012	saitama stadium 2002 , saitama	2 - 0	6 - 0	2014 fifa world cup qualification
8 june 2012	saitama stadium 2002 , saitama	3 - 0	6 - 0	2014 fifa world cup qualification
8 june 2012	saitama stadium 2002 , saitama	5 - 0	6 - 0	2014 fifa world cup qualification
6 february 2013	home 's stadium kobe , kobe	2 - 0	3 - 0	friendly
4 june 2013	saitama stadium 2002 , saitama	1 - 1	1 - 1	2014 fifa world cup qualification
19 june 2013	itaipava arena pernambuco , recife	1 - 0	3 - 4	2013 fifa confederations cup
14 august 2013	miyagi stadium , rifu	2 - 4	2 - 4	friendly
6 september 2013	nagai stadium , osaka	1 - 0	3 - 0	friendly
10 september 2013	international stadium yokohama , kanagawa	3 - 1	3 - 1	friendly
as of 6 september 2013	as of 6 september 2013	as of 6 september 2013	as of 6 september 2013	as of 6 september 2013

Baseline: keisuke honda played at the miyagi stadium , rifu on 10 october 2009 and on 6 september 2013
Ours: keisuke honda played in 11 competitions in the cup
`eq { count { filter_eq { all_rows ; competition ; cup } } ; 11 }` - substring matching

Interesting: none / e2e / both

7.6 Discussion

Overall, both automatic and manual evaluation showed that the proposed system is superior to the baseline model in terms of fidelity, and is not much worse in the interestingness aspect. Nevertheless, we argue that the current metrics for logical table-to-text generation are not entirely reliable. Apart from our main analysis presented above which showed only a weak correlation with the automatic metric, we have noticed the unwanted behaviour of both TaPEx and TaPas models in three cases:

1. Random shuffle: if the predictions are randomly shuffled so that they almost never match the input tables, both metrics output the score of > 0.5 thus predicting that over a half of predictions are actually entailed by the table. Therefore, the metrics are over-positive towards unfaithful statements.

2. Nonsense outputs: during our preliminary experiments, one model apparently converged to the incorrect local minimum and produced nonsense outputs that do no not correspond to any table at all (examples in Appendix A.9). However, TaPEx-acc metric is 96.0, whereas TaPas-acc is 89.5. The reason behind this behaviour is yet to be determined, but this accident shows the relative unreliability of these models and model-based evaluation in general.

3. Empty statements: the models can predict the "entailed" class for empty statements.

As we can see, both metrics, despite different architectures, output unexpected results in some cases making the whole evaluation setup compromised.

Our next concern is the BLEU score. Apart from overall BLEU criticism discussed in Section 3.2, the setup of BLEU-1/2/3 employed by Chen et al. (2020a) and all subsequent work has two major drawbacks:

1. Although the prediction is conducted on a subset of columns to avoid noise and mitigate the input length issue, the evaluation is done using all references for the given table, which are constructed on different column subsets. Thus, the generated sequence is compared with references that are possibly absolutely unrelated to the table-sentence pair in consideration. Additional references might inflate the resulting score.

2. Chen et al. (2020a) employ sentence-level BLEU and average it across all evaluated sentences. This contradicts a conventional BLEU estimation setup where BLEU is evaluated on the whole corpus. We have found that the averaged sentence-level scores are considerably stricter than a corpus-level metric.

Nevertheless, it is not possible to apply SacreBLEU evaluation neatly either since the data in the employed datasets is pre-tokenized. Unfortunately, SacreBLEU implementation awaits a detokenized input, which is not the case for the current study. We provide the SacreBLEU score but it is necessary to keep in mind that this metric is not entirely applicable to this work either if no additional postprocessing is done.

Overall, we express a concern about current evaluation metrics for this task and encourage sanity checks and manual analysis of the metric outputs. Nevertheless, we suppose that the model-based metrics, despite their black-box nature and quite non-controllable behaviour, are the most suitable evaluation method for this task due to its complexity.

8 Conclusion

In this work, we studied two questions: whether it is possible to select relevant columns and operations from the table without using the whole table, and whether symbolic reasoning helps to improve statement fidelity. We discuss our contributions and outline the directions of the future work.

8.1 Contributions

The contributions of this thesis are as follows:

1. We showed that it is possible to do sensible content selection on only aggregated data, which allows to apply this system to tables with multiple rows. We found that the model benefits from more detailed input, further pre-training, and generation with sampling. However, we found that this part in its current state lacks variability, generating several template types too often.
2. We showed that symbolic reasoning greatly increases fidelity of the final outputs. Nevertheless, it is important to note that, as with any rule-based system, this step required a considerable amount of resources and time, as well as extensive testing to alleviate possible logic and coding errors.
3. We made a full pipeline combining these two approaches thus approaching to the solution both problems we addressed: long excessive input and fidelity. Moreover, the resulting system has two important properties. The first one is the opportunity to explicitly control generation by further incorporating changes to the content selection step. The second one, especially relevant today, is the interpretability, since every generated statement is based on the formal representation with explicitly stated operations, columns, and values chosen with a symbolic approach.
4. We have analyzed the original datasets in depth and discussed their potential problems such as output postprocessing and data imbalance.
5. We have highlighted the problems of the automatic evaluation metrics, encouraging sanity checks and qualitative analysis of metric outputs in general.

8.2 Future Work

This work can be improved in several ways, some of which are relatively easy to achieve, and others are more conceptual and require a deep dive into further development. Future work directions are the following:

1. evaluate our system end-to-end on Logic2Text dataset;
2. replace our basic model for LF-to-text generation with a model from other available works, for instance, the one released by Xie et al. (2022);
3. try out larger models and see if further improvement is possible by scaling up;
4. explore older fidelity metrics discarded by us but actively used in other works, analyze their behaviour and discuss their advantages and problems;
5. improve the variability of content selection model outputs;
6. improve data processing to account for more column types encountered in the data.

Bibliography

Ewa Andrejczuk, Julian Martin Eisenschlos, Francesco Piccinno, Syrine Krichene, and Yasemin Altun. Table-to-text generation and pre-training with tabt5, 2022. URL `https://arxiv.org/abs/2210.09162`.

Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL `https://aclanthology.org/W05-0909`.

Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. Methods for exploring and mining tables on wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, IDEA '13, page 18–26, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450323291. doi: 10.1145/2501511.2501516. URL `https://doi.org/10.1145/2501511.2501516`.

Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

Wenhu Chen. Large language models are few(1)-shot table reasoners. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1120–1130, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. URL `https://aclanthology.org/2023.findings-eacl.83`.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online, July 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.708. URL `https://aclanthology.org/2020.acl-main.708`.

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

Wenqing Chen, Jidong Tian, Yitian Li, Hao He, and Yaohui Jin. De-confounded variational encoder-decoder for logical table-to-text generation. 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 5532–5542, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.430. URL `https://aclanthology.org/2021.acl-long.430`.

Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. Logic2Text: High-fidelity natural language generation from logical forms. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2096–2111, Online, November 2020c. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.190. URL `https://aclanthology.org/2020.findings-emnlp.190`.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. HiTab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1094–1110, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.78. URL `https://aclanthology.org/2022.acl-long.78`.

Julian Eisenschlos, Syrine Krichene, and Thomas Müller. Understanding tables with intermediate pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.27. URL `https://aclanthology.org/2020.findings-emnlp.27`.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 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, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.398. URL `https://aclanthology.org/2020.acl-main.398`.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, nov 2022. ISSN 0360-0300. doi: 10.1145/3571730. URL `https://doi.org/10.1145/3571730`. Just Accepted.

Zdeněk Kasner, Ekaterina Garanina, Ondrej Platek, and Ondrej Dusek. TabGenie: A toolkit for table-to-text generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 444–455, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-demo.42. URL `https://aclanthology.org/2023.acl-demo.42`.

Rémi Lebret, David Grangier, and Michael Auli. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1128. URL https://aclanthology.org/D16-1128.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 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, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Ao Liu, Haoyu Dong, Naoaki Okazaki, Shi Han, and Dongmei Zhang. Plog: Table-to-logic pretraining for logical table-to-text generation. *arXiv preprint arXiv:2205.12697*, 2022a.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. TAPEX: Table pre-training via learning a neural SQL executor. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=O50443AsCP.

Nafise Sadat Moosavi, Andreas Rücklé, Dan Roth, and Iryna Gurevych. Learning to reason for text generation from scientific tables. *arXiv preprint arXiv:2104.08296*, 2021. URL https://arxiv.org/abs/2104.08296.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. DART: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.37. URL https://aclanthology.org/2021.naacl-main.37.

Linyong Nan, Lorenzo Jaime Flores, Yilun Zhao, Yixin Liu, Luke Benson, Weijin Zou, and Dragomir Radev. R2D2: Robust data-to-text with replacement detection. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6903–6917, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.464. URL https://aclanthology.org/2022.emnlp-main.464.

Jason Obeid and Enamul Hoque. Chart-to-text: Generating natural language descriptions for charts by adapting the transformer model. In *Proceedings of the*

13th International Conference on Natural Language Generation, pages 138–147, Dublin, Ireland, December 2020. Association for Computational Linguistics. URL https://aclanthology.org/2020.inlg-1.20.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.89. URL https://aclanthology.org/2020.emnlp-main.89.

Yotam Perlitz, Liat Ein-Dor, Dafna Sheinwald, Noam Slonim, and Michal Shmueli-Scheuer. Diversity enhanced table-to-text generation via type control, 2023.

Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6319. URL https://aclanthology.org/W18-6319.

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press, 2000. doi: 10.1017/CBO9780511519857.

David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pages 318–362. 1987.

Swarnadeep Saha, Xinyan Velocity Yu, Mohit Bansal, Ramakanth Pasunuru, and Asli Celikyilmaz. Murmur: Modular multi-step reasoning for semi-structured data-to-text generation, 2022. URL https://arxiv.org/abs/2212.08607.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.704. URL https://aclanthology.org/2020.acl-main.704.

Chang Shu, Yusen Zhang, Xiangyu Dong, Peng Shi, Tao Yu, and Rui Zhang. Logic-consistency text generation from semantic parses. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4414–4426, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.388. URL `https://aclanthology.org/2021.findings-acl.388`.

Lya Hulliyyatus Suadaa, Hidetaka Kamigaito, Kotaro Funakoshi, Manabu Okumura, and Hiroya Takamura. Towards table-to-text generation with numerical reasoning. 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 1451–1465, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.115. URL `https://aclanthology.org/2021.acl-long.115`.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Sam Wiseman, Stuart Shieber, and Alexander Rush. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1239. URL `https://aclanthology.org/D17-1239`.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL `https://aclanthology.org/2020.emnlp-demos.6`.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. UnifiedSKG: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings* *of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.39. URL `https://aclanthology.org/2022.emnlp-main.39`.

Xueliang Zhao, Tingchen Fu, Lemao Liu, Lingpeng Kong, Shuming Shi, and Rui Yan. SORTIE: Dependency-aware symbolic reasoning for logical data-to-text generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11247–11266, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.715. URL `https://aclanthology.org/2023.findings-acl.715`.

Yilun Zhao, Linyong Nan, Zhenting Qi, Rui Zhang, and Dragomir Radev. ReasTAP: Injecting table reasoning skills during pre-training via synthetic reasoning examples. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9006–9018, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.615. URL `https://aclanthology.org/2022.emnlp-main.615`.

Yilun Zhao, Zhenting Qi, Linyong Nan, Lorenzo Jaime Flores, and Dragomir Radev. LoFT: Enhancing faithfulness and diversity for table-to-text generation via logic form control. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 554–561, Dubrovnik, Croatia, May 2023b. Association for Computational Linguistics. URL `https://aclanthology.org/2023.eacl-main.40`.

Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. Large language models are effective table-to-text generators, evaluators, and feedback providers, 2023c.

List of Figures

List of Tables

A Appendices

A.1 Attachments

We release additional materials along with the current thesis:

1. The code for pipeline training and inference;
2. The models for the two steps of our pipeline;
3. Auxiliary scripts for data processing and analysis;
4. Manual evaluation materials: app code, input data, raw responses;
5. The outputs of our best system on the LogicNLG development and test sets.

We have made the code and the system outputs available on GitHub[1], and have uploaded the models to Hugging Face Hub[2]. Moreover, we attach a `.zip` archive containing the fixed version of the code, outputs, auxiliary scripts, and manual evaluation materials.

[1] `https://github.com/kategerasimenko/LT2T`

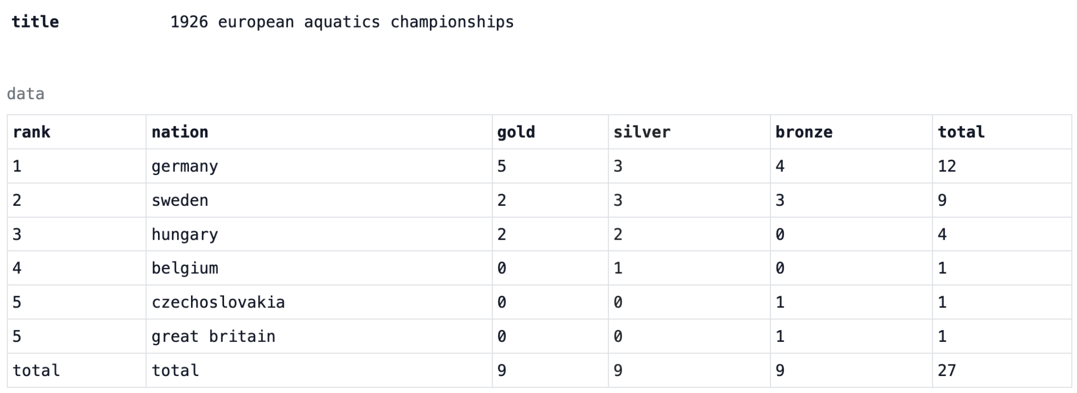
[2] Content selection: `https://huggingface.co/kategaranina/lt2t_content_selection`
LF-to-text: `https://huggingface.co/kategaranina/lt2t_lf_to_text`

A.2 Function Examples

function	LF	reference	source
count	eq { count { filter_eq { all_rows ; bronze ; 0 } } ; 2 }	two of these countries earned a total of 0 bronze medals .	dev 19, 2-15972223-1
only	and { only { filter_eq { all_rows ; largest ethnic group (2002) ; slovaks } } ; eq { hop { filter_eq { all_rows ; largest ethnic group (2002) ; slovaks } ; settlement } ; pivnice } }	pivnice is the only settlement in vojvodina with slovaks as the largest ethnic group .	train 1, 1-2562572-12
hop	eq { hop { nth_argmax { all_rows ; rounds ; 1 } ; team } ; phil parsons racing }	the team that participated in the highest number of rounds was phil parsons racing .	train 15, 2-1266602-2
and	and { eq { nth_max { all_rows ; goals against ; 1 } ; 333 } ; eq { hop { nth_argmax { all_rows ; goals against ; 1 } ; season } ; 1982 - 83 } }	with 378 goals , 1982-83 was the nova scotia voyageurs highest scoring season .	train 1241, 2-1166259-1
max	and { eq { max { all_rows ; capacity } ; 41040 } ; eq { hop { argmax { all_rows ; capacity } ; team } ; dinamo minsk } }	in the 1993 - 94 belarusian premier league , the venue with the highest capacity was minsk at 41040 .	train 369, 2-14744886-1
avg	round_eq { avg { all_rows ; weight (lb) } ; 212.14 }	for the 2009 - 10 pittsburgh panthers men 's basketball team , the average weight for the players was 212.14 .	train 50, 1-24925945-3
nth_max	eq { nth_max { all_rows ; no of episodes ; 3 } ; 135 }	the third-highest number of episodes in a show shown on farsi1 was 135 .	train 192, 1-28803803-1
argmax	eq { hop { argmax { all_rows ; earnings } ; player } ; lee trevino }	lee trevino had the highest earnings of any player in the 1996 senior pga tour .	train 28, 2-11621873-4
nth_argmax	eq { hop { nth_argmax { all_rows ; face value ; 4 } ; ecosystem } ; alpine tundra }	the alpine tundra ecosystem series of nature of america stamps has the fourth highest face value .	train 8339, 2-15635768-1
eq	eq { hop { nth_argmax { all_rows ; gold ; 2 } ; nation } ; argentina (arg) }	argentina (arg) recorded the 2nd highest number of gold in athletics at the 1951 pan american games .	dev 3, 2-10647806-3
not_eq	and { not_eq { hop { filter_eq { all_rows ; player ; bob jones } ; team } ; hop { filter_eq { all_rows ; player ; larry hutton } ; team } } ; and { eq { hop { filter_eq { all_rows ; player ; bob jones } ; team } ; minnesota twins } ; eq { hop { filter_eq { all_rows ; player ; larry hutton } ; team } ; los angeles dodgers } } }	bob jones was drafted by the minnesota twins and larry hutton was drafted by the los angeles dodgers in the 1966 major league baseball draft .	train 1816, 2-15667202-1
round_eq	round_eq { sum { all_rows ; points } ; 2 }	between 1952 and 1956 , elie bayol scored a total of 2 points .	train 6, 2-1228323-1
greater	greater { hop { filter_eq { all_rows ; nation ; puerto rico } ; bronze } ; hop { filter_eq { all_rows ; nation ; barbados } ; bronze } }	puerto rico recorded more bronze medals than barbados in athletics at the 1986 central american and caribbean games .	train 61, 2-10258265-3
diff	eq { diff { hop { filter_eq { all_rows ; title ; joke overload } ; original air date } ; hop { filter_eq { all_rows ; title ; end of the middle } ; original air date } } ; -7 }	the childrens hospital episode titled " joke overload " had an original air date that was 7 days before the original air date for the episode titled " end of the middle . " .	train 334, 1-28081876-4
filter_eq (substr)	eq { count { filter_eq { all_rows ; method ; submission } } ; 4 }	4 of jake o'brien 's fights have ended due to submission .	train 313, 2-15985163-2
filter_greater	eq { count { filter_less { filter_greater { all_rows ; population ; 50000 } ; altitude (mslm) ; 300 } } ; 2 }	in the province of turin , 2 of those with population more than 50000 have altitude (mslm) of less than 300 .	dev 85, 2-1449176-1
filter_greater_eq	eq { count { filter_greater_eq { all_rows ; goals for ; 50 } } ; 7 }	7 teams score 50 or more goals in the 1988 - 89 segunda división .	train 146, 2-12107896-2
filter_all	eq { count { filter_all { all_rows ; area served } } ; 6 }	there are 6 areas served of radio stations in the northern territory .	train 51, 2-14155573-2
all_eq	all_eq { all_rows ; home captain ; alec stewart }	alec stewart was the home captain all of england 's cricket test matches .	dev 89, 2-12410929-94
all_greater	all_greater { all_rows ; elevation (m) ; 1000 }	all of the mountains in norway have an elevation that is higher than 1000 meters .	train 1098, 2-12280396-1
all_greater_eq	all_greater_eq { all_rows ; rebounds ; 100 }	all of the top players in euroleague 2007 - 08 had 100 or more rebounds .	train 635, 2-16050349-2
most_eq (substr)	most_eq { all_rows ; result ; l }	the majority of games in the 2006 kansas city brigade season ended in losses for the kansas city brigade .	train 138, 2-11974088-1
most_not_eq	most_not_eq { all_rows ; qual 2 ; - }	most drivers of the 2004 centrix financial grand prix of denver had a qual 2 time .	train 874, 2-16789804-1
most_greater	most_greater { all_rows ; crowd ; 10000 }	most of the games on july 4 , 1931 had a crowd of over 10,000 .	train 254, 2-10789881-9
most_greater_eq	most_greater_eq { all_rows ; td 's ; 1 }	most of the players from tampa bay storm had at least one td in the 2007 season .	train 540, 2-11486671-4

Table A.1: Examples of all function types.

A.3 Table Examples

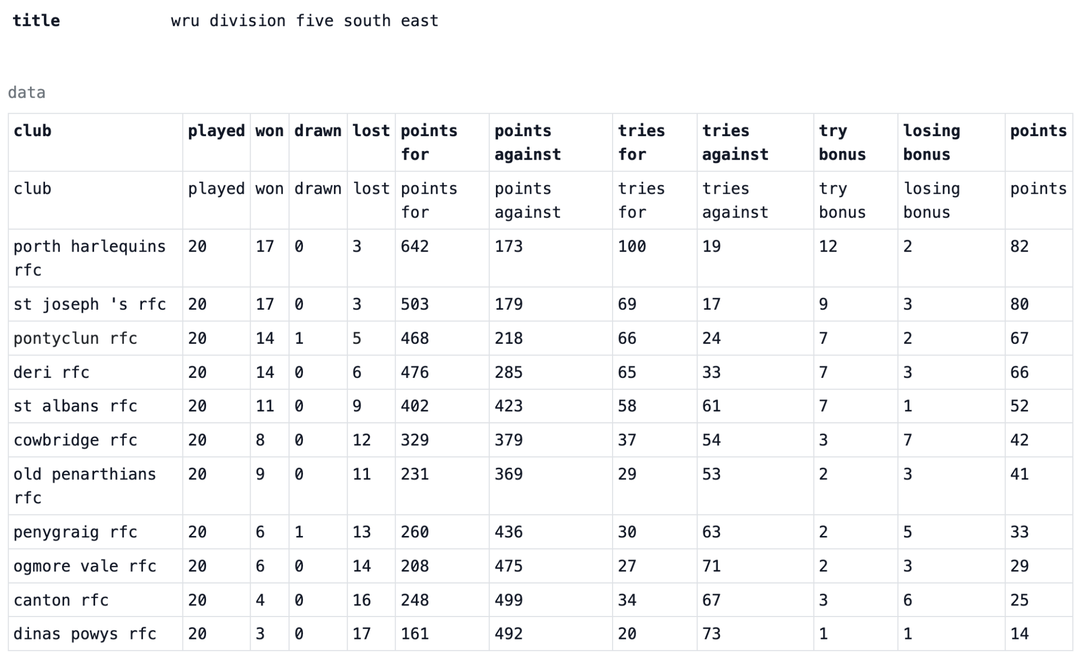


title	1926 european aquatics championships

data

rank	**nation**	**gold**	**silver**	**bronze**	**total**
1	germany	5	3	4	12
2	sweden	2	3	3	9
3	hungary	2	2	0	4
4	belgium	0	1	0	1
5	czechoslovakia	0	0	1	1
5	great britain	0	0	1	1
total	total	9	9	9	27

Figure A.1: Table with the last summing row. Logic2Text, train 5880, id `2-10636637-1`.

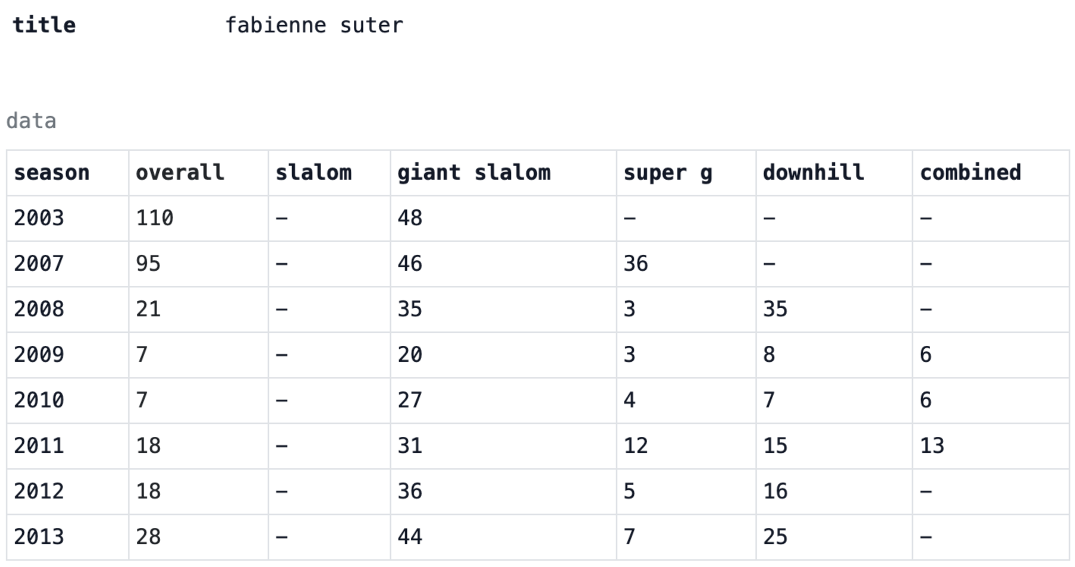


title	wru division five south east

data

club	**played**	**won**	**drawn**	**lost**	**points for**	**points against**	**tries for**	**tries against**	**try bonus**	**losing bonus**	**points**
club	played	won	drawn	lost	points for	points against	tries for	tries against	try bonus	losing bonus	points
porth harlequins rfc	20	17	0	3	642	173	100	19	12	2	82
st joseph 's rfc	20	17	0	3	503	179	69	17	9	3	80
pontyclun rfc	20	14	1	5	468	218	66	24	7	2	67
deri rfc	20	14	0	6	476	285	65	33	7	3	66
st albans rfc	20	11	0	9	402	423	58	61	7	1	52
cowbridge rfc	20	8	0	12	329	379	37	54	3	7	42
old penarthians rfc	20	9	0	11	231	369	29	53	2	3	41
penygraig rfc	20	6	1	13	260	436	30	63	2	5	33
ogmore vale rfc	20	6	0	14	208	475	27	71	2	3	29
canton rfc	20	4	0	16	248	499	34	67	3	6	25
dinas powys rfc	20	3	0	17	161	492	20	73	1	1	14

Figure A.2: Table with the duplicated header. Logic2Text, train 66, id `1-17625749-3`.

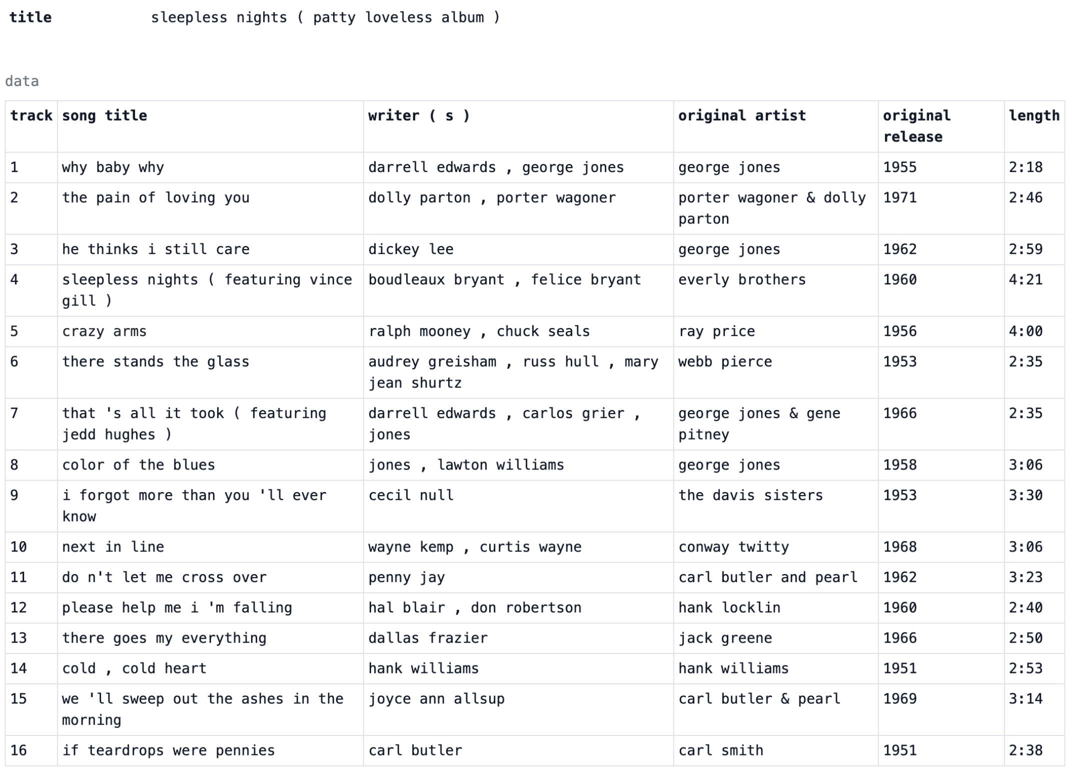


title fabienne suter

data

season	overall	slalom	giant slalom	super g	downhill	combined
2003	110	–	48	–	–	–
2007	95	–	46	36	–	–
2008	21	–	35	3	35	–
2009	7	–	20	3	8	6
2010	7	–	27	4	7	6
2011	18	–	31	12	15	13
2012	18	–	36	5	16	–
2013	28	–	44	7	25	–

Figure A.3: Table with empty values, represented as `-`. Column *slalom* is of type `empty`. Logic2Text, train 6244, id `2-16403980-1`.

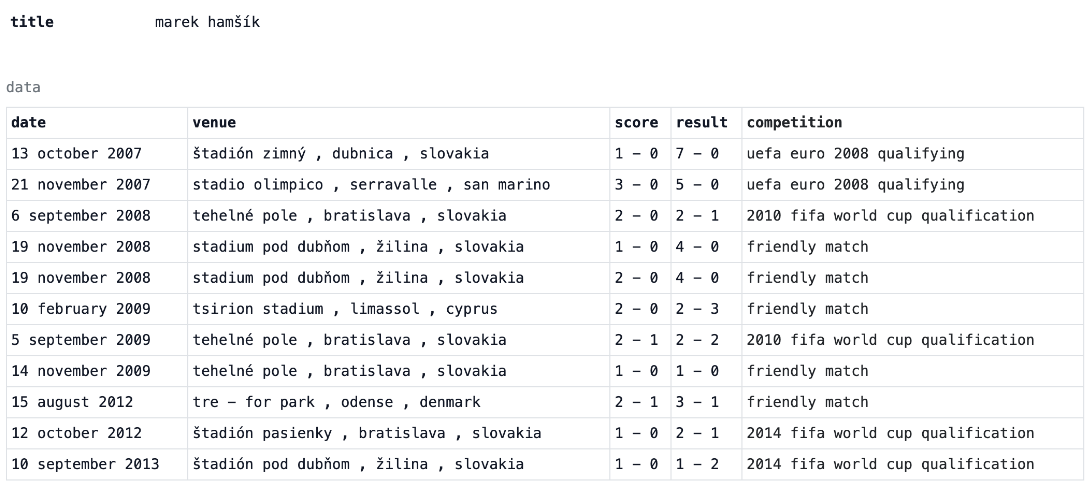


title sleepless nights (patty loveless album)

data

track	song title	writer (s)	original artist	original release	length
1	why baby why	darrell edwards , george jones	george jones	1955	2:18
2	the pain of loving you	dolly parton , porter wagoner	porter wagoner & dolly parton	1971	2:46
3	he thinks i still care	dickey lee	george jones	1962	2:59
4	sleepless nights (featuring vince gill)	boudleaux bryant , felice bryant	everly brothers	1960	4:21
5	crazy arms	ralph mooney , chuck seals	ray price	1956	4:00
6	there stands the glass	audrey greisham , russ hull , mary jean shurtz	webb pierce	1953	2:35
7	that 's all it took (featuring jedd hughes)	darrell edwards , carlos grier , jones	george jones & gene pitney	1966	2:35
8	color of the blues	jones , lawton williams	george jones	1958	3:06
9	i forgot more than you 'll ever know	cecil null	the davis sisters	1953	3:30
10	next in line	wayne kemp , curtis wayne	conway twitty	1968	3:06
11	do n't let me cross over	penny jay	carl butler and pearl	1962	3:23
12	please help me i 'm falling	hal blair , don robertson	hank locklin	1960	2:40
13	there goes my everything	dallas frazier	jack greene	1966	2:50
14	cold , cold heart	hank williams	hank williams	1951	2:53
15	we 'll sweep out the ashes in the morning	joyce ann allsup	carl butler & pearl	1969	3:14
16	if teardrops were pennies	carl butler	carl smith	1951	2:38

Figure A.4: Table with column types `rank` (*track*), `year` (*original release*), and `time` (*length*). Logic2Text, test 463, id `2-18424482-2`.

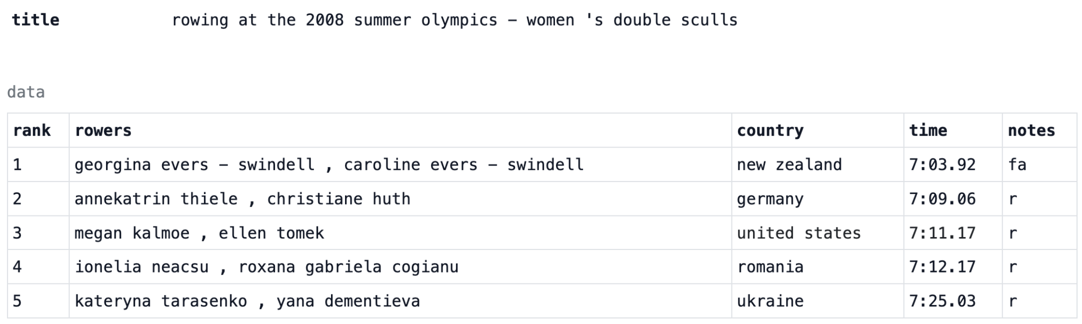


title marek hamšík

data

date	venue	score	result	competition
13 october 2007	štadión zimný , dubnica , slovakia	1 – 0	7 – 0	uefa euro 2008 qualifying
21 november 2007	stadio olimpico , serravalle , san marino	3 – 0	5 – 0	uefa euro 2008 qualifying
6 september 2008	tehelné pole , bratislava , slovakia	2 – 0	2 – 1	2010 fifa world cup qualification
19 november 2008	stadium pod dubňom , žilina , slovakia	1 – 0	4 – 0	friendly match
19 november 2008	stadium pod dubňom , žilina , slovakia	2 – 0	4 – 0	friendly match
10 february 2009	tsirion stadium , limassol , cyprus	2 – 0	2 – 3	friendly match
5 september 2009	tehelné pole , bratislava , slovakia	2 – 1	2 – 2	2010 fifa world cup qualification
14 november 2009	tehelné pole , bratislava , slovakia	1 – 0	1 – 0	friendly match
15 august 2012	tre – for park , odense , denmark	2 – 1	3 – 1	friendly match
12 october 2012	štadión pasienky , bratislava , slovakia	1 – 0	2 – 1	2014 fifa world cup qualification
10 september 2013	štadión pod dubňom , žilina , slovakia	1 – 0	1 – 2	2014 fifa world cup qualification

Figure A.5: Logic2Text, test 416, id `2-11895475-1`.

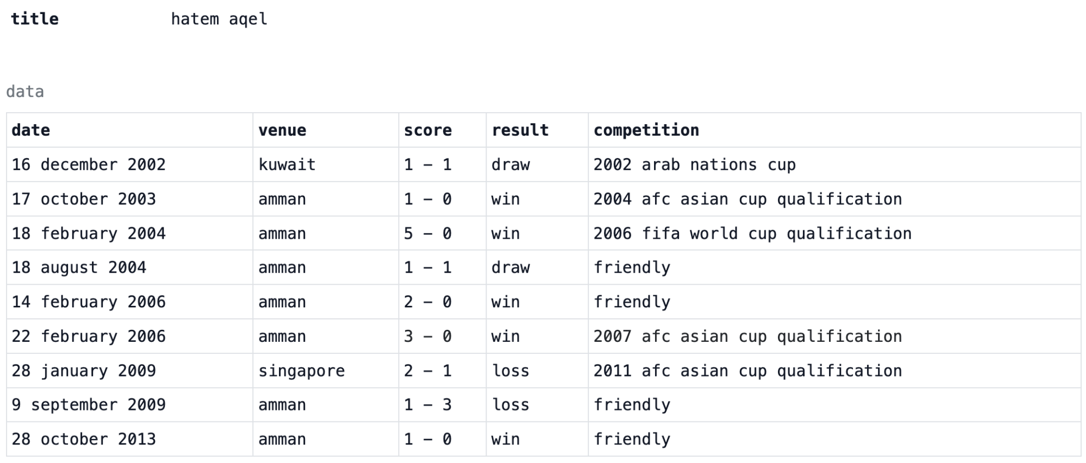


title rowing at the 2008 summer olympics – women 's double sculls

data

rank	rowers	country	time	notes
1	georgina evers – swindell , caroline evers – swindell	new zealand	7:03.92	fa
2	annekatrin thiele , christiane huth	germany	7:09.06	r
3	megan kalmoe , ellen tomek	united states	7:11.17	r
4	ionelia neacsu , roxana gabriela cogianu	romania	7:12.17	r
5	kateryna tarasenko , yana dementieva	ukraine	7:25.03	r

Figure A.6: Logic2Text, test 18, id `2-18662700-3`.

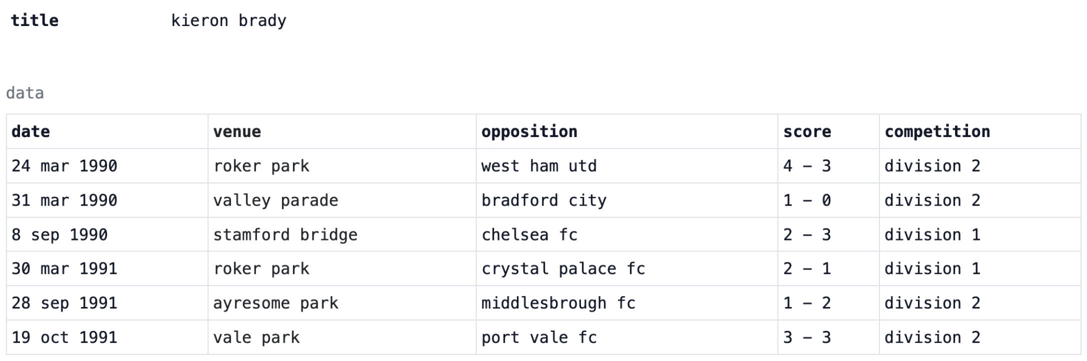


title hatem aqel

data

date	venue	score	result	competition
16 december 2002	kuwait	1 – 1	draw	2002 arab nations cup
17 october 2003	amman	1 – 0	win	2004 afc asian cup qualification
18 february 2004	amman	5 – 0	win	2006 fifa world cup qualification
18 august 2004	amman	1 – 1	draw	friendly
14 february 2006	amman	2 – 0	win	friendly
22 february 2006	amman	3 – 0	win	2007 afc asian cup qualification
28 january 2009	singapore	2 – 1	loss	2011 afc asian cup qualification
9 september 2009	amman	1 – 3	loss	friendly
28 october 2013	amman	1 – 0	win	friendly

Figure A.7: Logic2Text, test 32, id `2-11048203-1`.

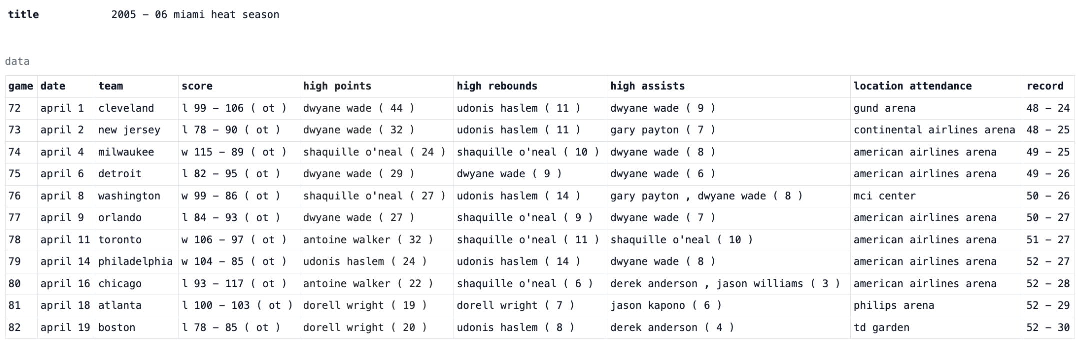


title kieron brady

data

date	venue	opposition	score	competition
24 mar 1990	roker park	west ham utd	4 – 3	division 2
31 mar 1990	valley parade	bradford city	1 – 0	division 2
8 sep 1990	stamford bridge	chelsea fc	2 – 3	division 1
30 mar 1991	roker park	crystal palace fc	2 – 1	division 1
28 sep 1991	ayresome park	middlesbrough fc	1 – 2	division 2
19 oct 1991	vale park	port vale fc	3 – 3	division 2

Figure A.8: Logic2Text, test 39, id 2-11788447-2.

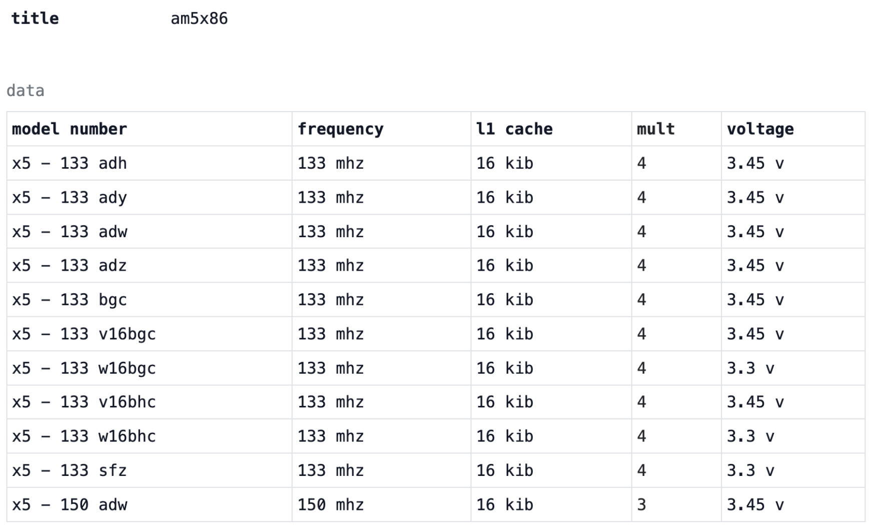


title 2005 – 06 miami heat season

data

game	date	team	score	high points	high rebounds	high assists	location attendance	record
72	april 1	cleveland	l 99 – 106 (ot)	dwyane wade (44)	udonis haslem (11)	dwyane wade (9)	gund arena	48 – 24
73	april 2	new jersey	l 78 – 90 (ot)	dwyane wade (32)	udonis haslem (11)	gary payton (7)	continental airlines arena	48 – 25
74	april 4	milwaukee	w 115 – 89 (ot)	shaquille o'neal (24)	shaquille o'neal (10)	dwyane wade (8)	american airlines arena	49 – 25
75	april 6	detroit	l 82 – 95 (ot)	dwyane wade (29)	dwyane wade (9)	dwyane wade (6)	american airlines arena	49 – 26
76	april 8	washington	w 99 – 86 (ot)	shaquille o'neal (27)	udonis haslem (14)	gary payton , dwyane wade (8)	mci center	50 – 26
77	april 9	orlando	l 84 – 93 (ot)	dwyane wade (27)	shaquille o'neal (9)	dwyane wade (7)	american airlines arena	50 – 27
78	april 11	toronto	w 106 – 97 (ot)	antoine walker (32)	shaquille o'neal (11)	shaquille o'neal (10)	american airlines arena	51 – 27
79	april 14	philadelphia	w 104 – 85 (ot)	udonis haslem (24)	udonis haslem (14)	dwyane wade (8)	american airlines arena	52 – 27
80	april 16	chicago	l 93 – 117 (ot)	antoine walker (22)	shaquille o'neal (6)	derek anderson , jason williams (3)	american airlines arena	52 – 28
81	april 18	atlanta	l 100 – 103 (ot)	dorell wright (19)	dorell wright (7)	jason kapono (6)	philips arena	52 – 29
82	april 19	boston	l 78 – 85 (ot)	dorell wright (20)	udonis haslem (8)	derek anderson (4)	td garden	52 – 30

Figure A.9: Logic2Text, test 40, id 2-13762472-8.

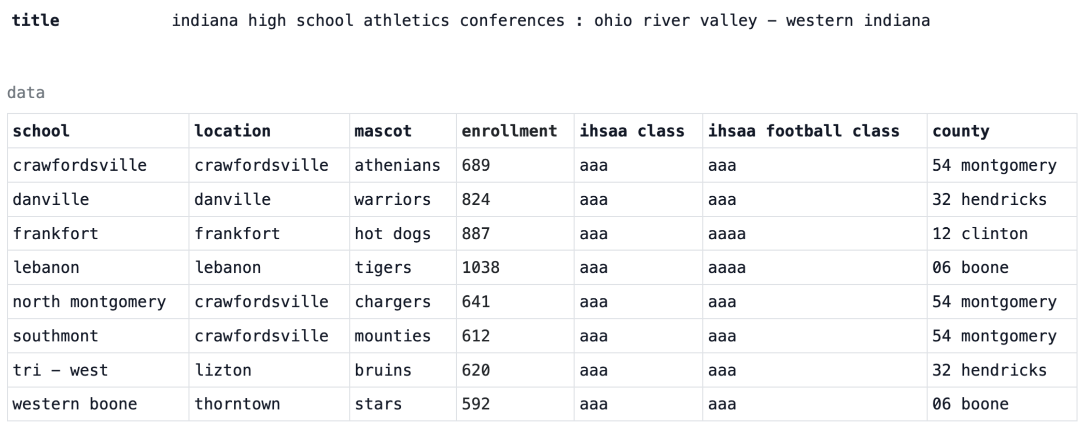


title am5x86

data

model number	frequency	l1 cache	mult	voltage
x5 – 133 adh	133 mhz	16 kib	4	3.45 v
x5 – 133 ady	133 mhz	16 kib	4	3.45 v
x5 – 133 adw	133 mhz	16 kib	4	3.45 v
x5 – 133 adz	133 mhz	16 kib	4	3.45 v
x5 – 133 bgc	133 mhz	16 kib	4	3.45 v
x5 – 133 v16bgc	133 mhz	16 kib	4	3.45 v
x5 – 133 w16bgc	133 mhz	16 kib	4	3.3 v
x5 – 133 v16bhc	133 mhz	16 kib	4	3.45 v
x5 – 133 w16bhc	133 mhz	16 kib	4	3.3 v
x5 – 133 sfz	133 mhz	16 kib	4	3.3 v
x5 – 150 adw	150 mhz	16 kib	3	3.45 v

Figure A.10: Logic2Text, test 83, id 2-167482-1.



title	indiana high school athletics conferences : ohio river valley – western indiana

data

school	location	mascot	enrollment	ihsaa class	ihsaa football class	county
crawfordsville	crawfordsville	athenians	689	aaa	aaa	54 montgomery
danville	danville	warriors	824	aaa	aaa	32 hendricks
frankfort	frankfort	hot dogs	887	aaa	aaaa	12 clinton
lebanon	lebanon	tigers	1038	aaa	aaaa	06 boone
north montgomery	crawfordsville	chargers	641	aaa	aaa	54 montgomery
southmont	crawfordsville	mounties	612	aaa	aaa	54 montgomery
tri – west	lizton	bruins	620	aaa	aaa	32 hendricks
western boone	thorntown	stars	592	aaa	aaa	06 boone

Figure A.11: Logic2Text, test 148, id `2-18974097-6`.

A.4 Model parameters

In this section, we list all hyperparameters used for training and generation and fixed during the experiments. All unlisted possible parameters are equal to defaults defined in `transformers` library of version `4.25.1` (Wolf et al., 2020).

Training data for all models was converted to ASCII using Unidecode library[3] (e.g., $\check{s} \rightarrow s$) in order to avoid OOV tokens in the input data as well as predictions. This conversion did not affect evaluation data.

All training was conducted on a single NVIDIA A100 40GB GPU card. We thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Hábrók high performance computing cluster.

A.4.1 Baseline

- `t5-base` model;
- epochs: 20;
- batch size: 16;
- learning rate: 1e-04;
- beams: 3.

A.4.2 Content Selection

Pre-training:

- `t5-base` model;
- 1 epoch;
- batch size: 16;
- learning rate: 1e-04;
- minimizing eval loss for evaluation and best checkpoint selection.

Fine-tuning:

- `t5-base` model;
- epochs: 20;
- batch size: 8;
- learning rate: 1e-04;
- top-k sampling with $k = 50$ during generation, if not stated otherwise.

[3]`https://github.com/avian2/unidecode`

A.4.3 LF-to-Text

- `t5-base` model;
- epochs: 30;
- batch size: 8;
- learning rate: 2e-05;
- beams: 3.

A.5 Template Filling Functions Details

Below we provide several comments about the non-trivial implementation details of the functions described in the function implementation table.

E1 Each cell has two types of values: an original raw value and a processed one obtained by parsing described in Section 6.2. If the placeholder is filled with a concrete value rather than a derivation or aggregation, raw cell value is used.

E2 If the function is used as a child of `eq` function (`(nth_)max/min`, partially `hop`), it returns raw value as well, which is later used to fill the placeholder in `eq` function. In the example below, `max` returns *27 july 2011*, which is a raw value from the table rather than a `datetime` object.

---

train 982, `2-17310356-2`

```
eq { max { all_rows ; took office } ; 27 july 2011 }
```

the latest date that a prime minister took office in the berlusconi iv cabinet was in july of 2011 .

E3 All boolean functions can be considered dummy in terms of execution. They return `True` and their truth value is verified during value options selection and consistency checks.

E4 Column types which allow at least some numerical operations are `num`, `rank`, `year`, `time`, `date`. We will further refer to this set of types as "numerical".

E5 When working with numerical columns, all `None` values are discarded. Nevertheless, they are taken into account when concrete values are considered (e.g. for `filter_eq` function).

E6 For all `*_greater` and `*_less` functions, column types `year`, `time`, `date` are not included despite being numeric because the rounding algorithm (described in Section 6.4.4 for numbers) is non-trivial to devise for them.

E7 Aggregating functions `avg` and `sum` do not work for types `date`, `year`, and `rank` because aggregations over these types do not convey any meaning.

E8 Values of type `rank` will not produce sensible generations for `most_*` and `all_*` functions (consider making statements about most values being larger than rank 2) and we do not list this type as valid for the above-mentioned functions.

E9 Functions `diff`, `greater`, and `less` can accept only numerical values but they have no direct restriction on column type since they accept only computed values. Therefore, their executability depends on the executability of their children, which must return numerical values.

E10 For all `nth_*` functions, we limit the option space by the first half of values which are sorted in descending (for `max`) or ascending (for `min`) order. It contradicts with some examples in the training data but corresponds more to common sense and expected function behaviour. In the example below, the table has 7 rows and our system suggests top-3 values, whereas the reference suggests taking the 5th row.

---

train 4895, `1-29574579-1`

Input:

```
eq { hop { nth_argmax { all_rows ; uk viewers ( million ) ; X } ; title } ;
episode X }
```

Suggested options:

```
eq { hop { nth_argmax { all_rows ; uk viewers ( million ) ; 1 } ; title } ;
episode 1 }
```

```
eq { hop { nth_argmax { all_rows ; uk viewers ( million ) ; 2 } ; title } ;
episode 2 }
```

```
eq { hop { nth_argmax { all_rows ; uk viewers ( million ) ; 3 } ; title }
episode 3 }
```

Reference:

```
eq { hop { nth_argmax { all_rows ; uk viewers ( million ) ; 5 } ; title }
episode 4 }
```

episode 4 had the 5th highest amount of uk viewers of all outcasts episodes .

E11 For `nth_argmax` and `nth_argmin` we introduce a consistency check for `and` function since one of the patterns in the data is `and` with two children, with the first one containing `nth_max / min`. `n` must be the same in both parts:

---

train 1241, `2-1166259-1`

```
and { eq { nth_max { all_rows ; goals against ; 1 } ; 333 } ; eq { hop {
nth_argmax { all_rows ; goals against ; 1 } ; season } ; 1982 - 83 } }
```

with 378 goals , 1982-83 was the nova scotia voyageurs highest scoring season .

E12 Function `eq` is the only one which has two versions of argument sets, and they need to be handled separately. The most frequent one has a value as a second argument, which needs to be filled. In this example, the filled value is *lee trevino*:

---

train 28, `2-11621873-4`

```
eq { hop { argmax { all_rows ; earnings } ; player } ; lee trevino }
```

lee trevino had the highest earnings of any player in the 1996 senior pga tour .

The second version consists of two LFs as both arguments, and this function only needs to be executed and return `True`. We run a consistency check for this structure.

---

train 441, `2-18620528-14`

```
eq { hop { filter_eq { all_rows ; class ; a - 4 } ; quantity made } ; hop {
filter_eq { all_rows ; class ; a - 3 } ; quantity made } }
```

the northern pacific railway locomotive 's class a - 4 and class a-3 had the same amount of quantity made .

E13 All `most_*` functions operate with strict majority, which is not always in line with the training data but aligns better with expected function behaviour. In the following example, the table has 6 lines, and *1948* occcurs 3 times, which is not strictly `most`, and our system does not fill the template:

---

train 343, `1-1342198-17`

```
most_eq { all_rows ; first elected ; 1948 }
```

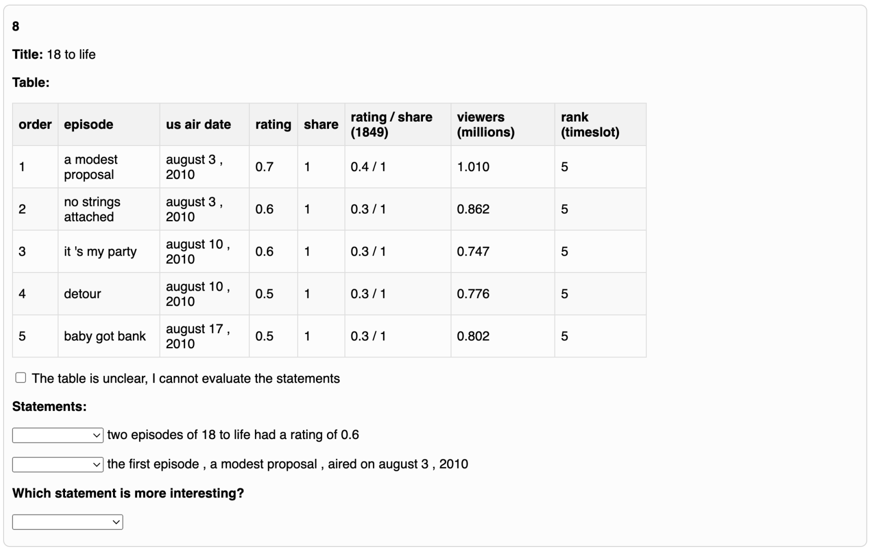
most of the incumbents in the 1950 us house of representatives elections in kentucky were first elected in 1948 .

A.6 Manual Evaluation Guidelines

Instruction:

> You are given 25 tables, each with **two statements**. Each table has a **title**.
>
> First, evaluate **whether each statement is logically correct** based on the table. Evaluate only formal logical correctness at this step.
> If the sentence doesn't make sense in relation to the table or is just not fluent, so you cannot understand it and make any judgements about its logical correctness, select "nonsense".
> If the sentence is mostly correct but some details in it are not, or the reported information is not complete in relation to the table, select "partially correct". Some examples of partially correct statements are non-critical misinterpretation of the columns or listing several values correctly but skipping other eligible ones, making the statement misleading.
> Missing diacritics and lowercase should not be counted as errors.
>
> Next, decide **which of the two statements is more interesting** (or both are equally interesting or none of them is). An interesting fact about the table looks natural in the context of the table and you could meet it in a report or a news article. You could potentially report it yourself if you were to describe the table.
> If the sentence is not logically correct, evaluate its interestingness as if it were correct.
>
> If you don't understand the table and thus cannot make any judgements about the statements, put a tick on "**Table is unclear**".

An example of a question in the web app:



8

Title: 18 to life

Table:

order	episode	us air date	rating	share	rating / share (1849)	viewers (millions)	rank (timeslot)
1	a modest proposal	august 3 , 2010	0.7	1	0.4 / 1	1.010	5
2	no strings attached	august 3 , 2010	0.6	1	0.3 / 1	0.862	5
3	it 's my party	august 10 , 2010	0.6	1	0.3 / 1	0.747	5
4	detour	august 10 , 2010	0.5	1	0.3 / 1	0.776	5
5	baby got bank	august 17 , 2010	0.5	1	0.3 / 1	0.802	5

☐ The table is unclear, I cannot evaluate the statements

Statements:

two episodes of 18 to life had a rating of 0.6

the first episode , a modest proposal , aired on august 3 , 2010

Which statement is more interesting?

Figure A.12: An example of a question in the manual evaluation web app.

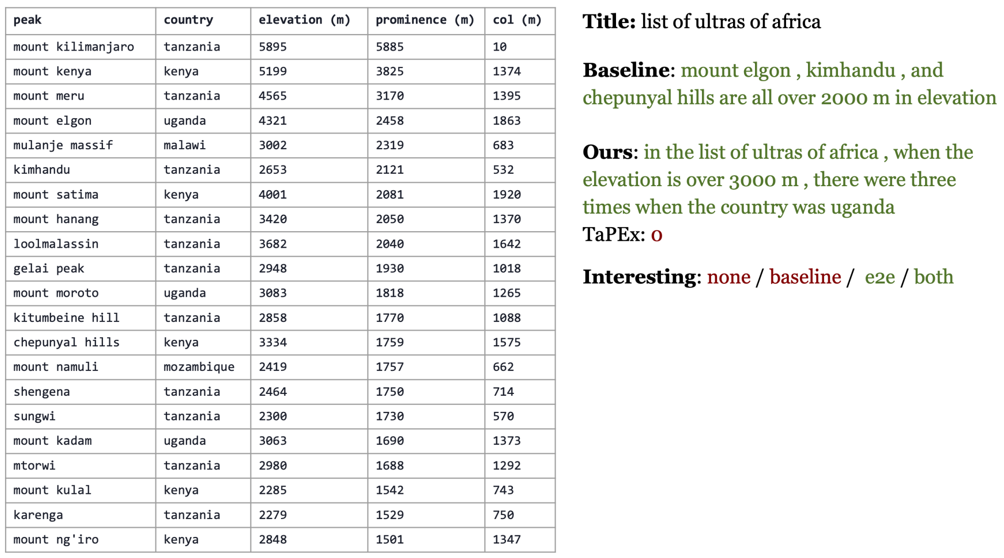
A.7 Correctness Evaluation per Annotator

category	system	Ann 1	Ann 2	Ann 3	Ann 4	Ann 5	avg
nonsense	baseline	11.1	12.5	10.7	13.4	5.0	10.5
	ours	6.1	6.3	13.1	6.2	5.0	7.3
incorrect	baseline	31.3	36.5	31.0	35.1	36.0	34.0
	ours	7.1	11.5	11.9	13.4	12.0	11.2
part_correct	baseline	16.2	12.5	14.3	15.5	12.0	14.1
	ours	4.0	7.3	4.8	6.2	5.0	5.5
correct	baseline	41.4	38.5	44.0	36.1	47.0	41.4
	ours	82.8	75.0	70.2	74.2	78.0	76.1
TaPEx corr for ours, p-value		0.216	0.459	0.512	0.298	0.028	0.063

Table A.2: Proportions of different correctness categories per system and per annotator. Last row: p-values of Fisher exact test between the manual evaluation and TAPEX metric on the outputs of our system - per annotator and on averaged data.

A.8 Wrong TaPEx Predictions

Example below illustrates the erroneous TAPEX output, which has predicted absence of entailment for the correct statement (LogicNLG, test 3976, id `2-18947170-9`):



peak	country	elevation (m)	prominence (m)	col (m)
mount kilimanjaro	tanzania	5895	5885	10
mount kenya	kenya	5199	3825	1374
mount meru	tanzania	4565	3170	1395
mount elgon	uganda	4321	2458	1863
mulanje massif	malawi	3002	2319	683
kimhandu	tanzania	2653	2121	532
mount satima	kenya	4001	2081	1920
mount hanang	tanzania	3420	2050	1370
loolmalassin	tanzania	3682	2040	1642
gelai peak	tanzania	2948	1930	1018
mount moroto	uganda	3083	1818	1265
kitumbeine hill	tanzania	2858	1770	1088
chepunyal hills	kenya	3334	1759	1575
mount namuli	mozambique	2419	1757	662
shengena	tanzania	2464	1750	714
sungwi	tanzania	2300	1730	570
mount kadam	uganda	3063	1690	1373
mtorwi	tanzania	2980	1688	1292
mount kulal	kenya	2285	1542	743
karenga	tanzania	2279	1529	750
mount ng'iro	kenya	2848	1501	1347

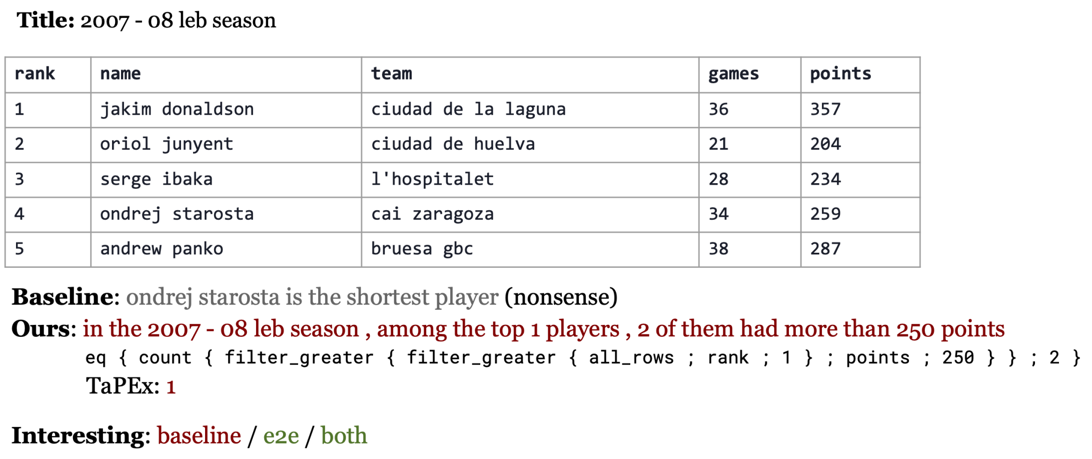
Title: list of ultras of africa

Baseline: mount elgon , kimhandu , and chepunyal hills are all over 2000 m in elevation

Ours: in the list of ultras of africa , when the elevation is over 3000 m , there were three times when the country was uganda
TaPEx: 0

Interesting: none / baseline / e2e / both

The following example illustrates another TAPEX error, the erroneous generation is marked as entailed by the table (LogicNLG, test 1859, id `2-16176685-4`):



Title: 2007 - 08 leb season

rank	name	team	games	points
1	jakim donaldson	ciudad de la laguna	36	357
2	oriol junyent	ciudad de huelva	21	204
3	serge ibaka	l'hospitalet	28	234
4	ondrej starosta	cai zaragoza	34	259
5	andrew panko	bruesa gbc	38	287

Baseline: ondrej starosta is the shortest player (nonsense)
Ours: in the 2007 - 08 leb season , among the top 1 players , 2 of them had more than 250 points

```
eq { count { filter_greater { filter_greater { all_rows ; rank ; 1 } ; points ; 250 } } ; 2 }
```

TaPEx: 1

Interesting: baseline / e2e / both

A.9 Nonsense Outputs

Two examples of nonsense outputs produced by the incorrectly trained model with TaPEx predictions:

Title: black ice (album)

country	date	label	format	catalogue no
europe	17 october 2008	columbia	cd , double lp	88697392232
australia	18 october 2008	sony music	cd	88697392382
united kingdom	20 october 2008	columbia	cd , double lp	88697392232
united kingdom	1 december 2008	columbia	cd (limited edition steel - box)	88697417452
united states	20 october 2008	columbia	cd	88697338292
japan	22 october 2008	sony music	cd	sicp - 2055
germany	5 december 2008	columbia	cd (limited edition steel - box)	886974174523
global (itunes)	19 november 2012	columbia	digital download	88697338292

Selected columns	Generated	TaPEx output
country	the average number of Point for a team in the top 5 ranking was 23.83	1
country, date	the average number of Point for a team in the top 5 ranking was 23.2	1
country, date, label	the average number of Point for a team in the top 5 ranking was 23.83	1
format	the average number of Point for a team in the top 5 ranking was 23.83	1

Figure A.13: LogicNLG, test 0, id `2-18424778-6`.

Title: utah jazz all - time roster

player	nationality	position	years for jazz	school / club team
adrian dantley	united states	guard - forward	1979 - 86	notre dame
brad davis	united states	guard	1979 - 80	maryland
darryl dawkins	united states	center	1987 - 88	maynard evans hs
paul dawkins	united states	guard	1979 - 80	northern illinois
greg deane	united states	guard	1979 - 80	utah
james donaldson	united states	center	1993 , 1994 - 95	washington state
john drew	united states	guard - forward	1982 - 85	gardner - webb
john duren	united states	guard	1980 - 82	georgetown

Selected columns	Generated	TaPEx output
player, years for jazz	the average number of Tackle for a member of the 1902 Michigan Football Wolverine was 14.5	1
player	the average number of Tackle for a member of the 1902 Michigan Football Wolverine was 14.5	1
position, years for jazz	the average number of Tackle for a member of the 1902 Michigan Football Wolverine was 14.5	1

Figure A.14: LogicNLG, test 25, id `2-11545282-4`.