DART: Open-Domain Structured Data Record to Text Generation

Linyong Nan[1] **Dragomir Radev**[1,2] **Rui Zhang**[3] **Amrit Rau**[1] **Abhinand Sivaprasad**[1]
Chiachun Hsieh[4] **Xiangru Tang**[1] **Aadit Vyas**[1] **Neha Verma**[1] **Pranav Krishna**[5]
Yangxiaokang Liu[1] **Nadia Irwanto**[1] **Jessica Pan**[1] **Faiaz Rahman**[1] **Ahmad Zaidi**[1]
Mutethia Mutuma[1] **Yasin Tarabar**[1] **Ankit Gupta**[1] **Tao Yu**[1] **Yi Chern Tan**[1]
Xi Victoria Lin[2*] **Caiming Xiong**[2] **Richard Socher**[2] **Nazneen Fatema Rajani**[2]
[1] Yale University [2] Salesforce Research [3] Penn State University
[4] The University of Hong Kong [5] MIT
{linyong.nan, dragomir.radev}@yale.edu, rmz5227@psu.edu, nazneen.rajani@salesforce.com

Abstract

We present **DART**, an open domain structured **DA**ta-**R**ecord-to-**T**ext generation dataset with over 82k instances (DARTs). Data-to-text annotations can be a costly process, especially when dealing with tables which are the major source of structured data and contain nontrivial structures. To this end, we propose a procedure of extracting semantic triples from tables that encodes their structures by exploiting the semantic dependencies among table headers and the table title. Our dataset construction framework effectively merged heterogeneous sources from open domain semantic parsing and spoken dialogue systems by utilizing techniques including tree ontology annotation, question-answer pair to declarative sentence conversion and predicate unification, all with minimum post-editing. We present systematic evaluation on DART as well as new state-of-the-art results on WebNLG 2017 to show that DART (1) poses new challenges to existing data-to-text datasets and (2) facilitates out-of-domain generalization. Our data and code can be found at https://github.com/Yale-LILY/dart.

1 Introduction

Automatically generating textual descriptions from structured data improves the accessibility of knowledge bases to lay users. Such applications include explaining data records to non-experts (Cawsey et al., 1997), writing sports news (Chen and Mooney, 2008), summarizing information in multiple documents (Fan et al., 2019), and generating dialogue responses (Wen et al., 2015).

While significant progress has been made in this field, there are still several issues with existing Data-to-Text datasets. First, they adopt a flat ontology structure of the data, such as slot-value pairs for data records (Lebret et al., 2016; Novikova et al., 2017b) or flat schema for tables (Wiseman et al., 2017; Chen et al., 2020a; Parikh et al., 2020). This flat structure is not powerful enough to encode rich semantic relationships in the ontology of the structured data, especially tables, whose representation can be further improved with these semantic knowledge. Second, some of the datasets only focus on a small number of domains or knowledge graphs, therefore providing limited number of predicates and data ontologies. For example, E2E (Novikova et al., 2017b) on restaurants and WebNLG (Gardent et al., 2017) on 15 categories from DBPedia. Furthermore, some of them only have loose alignments between data input and sentence due to the nature of the task (Wiseman et al., 2017) and the automatic generation procedure (Vougiouklis et al., 2018; Elsahar et al., 2018).

To address some of these issues and to encourage further research in natural language generation from structured data, we introduce **DART**, a large and open-domain structured **DA**ta-**R**ecord-to-**T**ext generation corpus. The goal of DART is to harvest the diverse predicates occurred in Wikipedia tables, which is significantly richer than those defined in the domain specific ontologies E2E and WebNLG were built on (Table 2). We also introduce a novel tree ontology annotation approach on tables, which converts a flat table schema into a tree structured semantic frame. The tree ontology reflects the core and auxiliary relations in the table schema, and naturally occurs across many domains. As a result, DART provides high-quality sentence annotations to tree structured semantic frames extracted from various data sources, including WikiSQL (Zhong et al., 2017) and WikiTableQuestions (Pasupat and Liang, 2015), two open-domain question answering datasets, as well as E2E (Novikova et al., 2017b) and WebNLG (Gardent et al., 2017) (Figure 1). We evaluated several state-of-the-art data-to-text models on DART, and found that while these models achieve impressive performance on domain-specific datasets, their performance suffers

*Now at Facebook AI.

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 432–447
June 6–11, 2021.

on DART due to its open-domain nature and richer semantic structures.

Our contributions are as follows. (1) We present a large and open-domain corpus for structured data record to text generation, annotated with tree ontologies converted from the table. This *hierarchical* input differentiates our corpus from existing data-to-text corpora. (2) We benchmark several state-of-the-art data-to-text models to show that DART introduces new generalization challenges. (3) We demonstrate that using DART for data augmentation improves the performance of existing models on the WebNLG 2017 dataset. We expect the results to generalize to other data-to-text datasets given the open-domain nature of DART.

2 DART Data Collection

As shown in Figure 1, DART is constructed from three different sources: (1) human annotation on Wikipedia tables from two table semantic parsing and question answering datasets WikiSQL and WikiTableQuestions (§ 2.1), (2) automatic conversion of questions in WikiSQL to declarative sentences (§ 2.2), and (3) incorporation of existing datasets including WebNLG 2017 and Cleaned E2E (§ 2.3). After collecting the ⟨triple-set, sentence⟩ pairs from various data sources, we manually canonicalized the predicates and show that DART covers a broad range of topics (§ 2.4). Finally, we discuss the data split in § 2.5.

2.1 Tree Ontology and Sentence Annotation on Tables

Tables are a major source of structured data that contain a wealth of information complementary to text and knowledge graphs. We aim to collect ⟨triple-set, sentence⟩ pairs from open-domain Wikipedia tables. However, table schema are flat, making them not directly usable for building subject-predicate-object triples to capture rich relationships in the data.

As shown in Figure 2, we propose a two-stage annotation process that involves two groups of annotators: internal annotators and Amazon Mechanical Turk[1] workers. In the first stage, skilled internal annotators specify the parent of every column header to construct a tree-structured ontology for each table. In the second stage, both internal and external annotators provide a sentential description of the highlighted cells in a row that are automatically-chosen based on the ontology.

Tree Ontology Annotation For each column in a given table, our internal annotators labeled its ontological parent. In Figure 2, for example, the annotator would provide the sequence {NULL, TEAM, STADIUM, STADIUM, TEAM} as the parent of each column — column TEAM has no parent, STADIUM has parent TEAM, and so on. In many cases, the relationship between a parent column and its child column can be conceptualized as a "has-a" relationship. For tables that are malformed or have duplicate or missing column names (as shown in Figure 5 of the Appendix), annotators either changed or added appropriate column names in order to fit these patterns. For each table we generate an ontology tree whose root is always [TABLECONTEXT]. This root node either has (1) one child node [TITLE] in the cases where the table title is the subject of entire table, or (2) column header node(s) and a [TITLE] node as children, as shown in Figure 2. This is because in some tables, the table title itself is more appropriate to be the root of the ontology tree (example shown in Figure 6 of the Appendix). In these cases, annotators assigned the special token [TITLE] as the parent of the relevant column nodes. For other tables, title usually provides important context for understanding the table's rows (example shown in Figure 7 of the Appendix). In such cases, [TITLE] is made a child of [TABLECONTEXT] together with the column headers that are appropriate.

We evaluate the quality of the initial tree ontology annotation and made corrections with the following procedure: (1) reject and request corrections from the original annotators if the provided ontology is disconnected or contains a cycle, (2) verify that all column headers appear as a node in the tree. For many tables, the determination of an ontology is a subjective process with many "correct" answers - for example, swapping the positions of TEAM and CITY in the tree in Figure 2 produces an equally valid ontology for the referenced table. If there are multiple ways to construct an ontology based on annotators' decisions of attribute relationships among column headers, we manually unify the annotations for similar tables (for examples, tables about athletes in different sports). The ontologies exhibit a great deal of structural variety. Relevant statistics are summarized in Table 7 and Figure 3 of the Appendix.

[1] https://www.mturk.com/

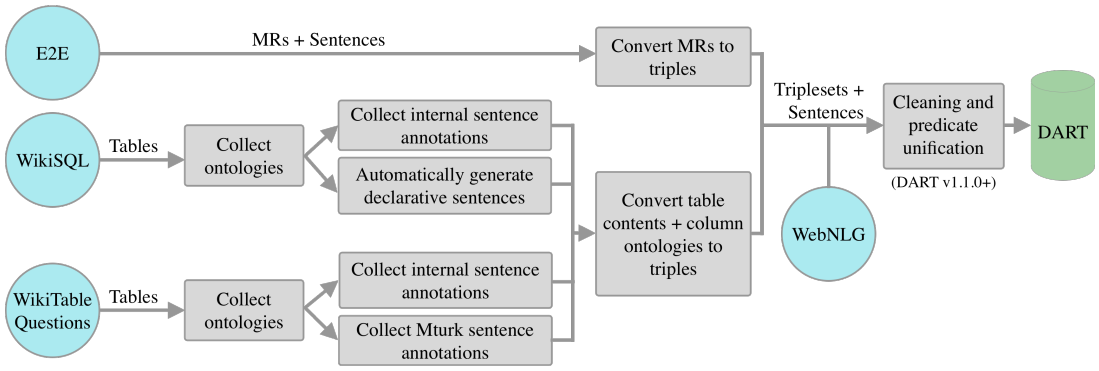

Figure 1: DART data collection pipeline. MR: Meaning Representation.

	Input Unit	Examples	Vocab Size	Words per SR	Sents per SR	Tables
WikiTableText	Row	13,318	—	13.9	1.0	4,962
LogicNLG	Table	37,015	122K	13.8	1.0	7,392
ToTTo	Highlighted Cells	136,161	136K	17.4	1.0	83,141
DART	Triple Set	82,191	33.2K	21.6	1.5	5,623

Table 1: DART compared with other open-domain table-to-text datasets. DART takes triple sets as input by incorporating the ontology of table headers and title, and its surface realizations tend to be longer with more than single sentence verbalization. SR: Surface Realization.

	DART: 62,659 train / 6,980 dev / 12,552 test					
	WikiTableQuestions		WikiSQL		WebNLG	Cleaned E2E
	Internal	MTurk	Internal	Declarative		
Domains	Wikipedia (open-domain)				15 DBPedia Categories	Restaurants
Unique Predicates	1,950	1,403	493	2,008	347	7
Unique Triples	13,505	5,541	1,648	7,787	3,220	946
Tripleset-Sentence Pairs	4,902	2,120	772	4,204	27,731	42,462
Triples per Tripleset (min, med, max)	1, 3, 10	1, 3, 7	1, 2, 7	1, 2, 10	1, 3, 7	1, 4, 7
Vocab Size	13.4K	8.9K	3.0K	10.7K	8.0K	3.0K
Words per SR	15.2	16.5	14.0	12.6	22.5	22.9
Sentences per SR	1.0	1.1	1.0	1.0	1.4	1.6

Table 2: Statistics of DART decomposed by different collection methods. DART exhibits a great deal of topical variety in terms of the number of unique predicates, the number of unique triples, and the vocabulary size.

Connected Component Extraction After we annotated the ontology, we automatically choose a subset of cells for a selected table row to form the triple set. Randomly selecting cells leads to poor quality annotation as the selected data could lack a subject, lack cohesion, or would require information not encoded in the ontology to form a coherent sentence. For example, in Figure 2, if only two nodes CITY and CAPACITY were highlighted then a coherent sentence cannot be produced as there is no direct logical relationship (functional dependency) between them. To solve these issues, instead of randomly selecting cells in a row, we extract connected components from the ontology.

The extracted components have two controllable properties: size and shape. To create variation in size, we randomly sampled between $[2, 5]$. The shape is determined by two numbers: the number of sibling node pairs and parent-child node pairs. Increasing the number of sibling node pairs creates a wider tree, while increasing the latter creates a deeper tree. We created a sliding scale between width and depth using an expansion parameter, p. We recursively visit a node if it has children with probability p and otherwise move to a sibling if it exists. If $p = 1$, the search becomes a DFS and if $p = 0$, it becomes BFS. We found that randomly selecting p from 0.5 to 0.7 created a reasonable variation in extracted component shapes. This ensures the balance between breadth and depth of ontology coverage of the selected cells, therefore ensuring the quality of the sentence annotation.

Sentence Annotation Given the table, title, and connected highlighted cells of a row, annotators

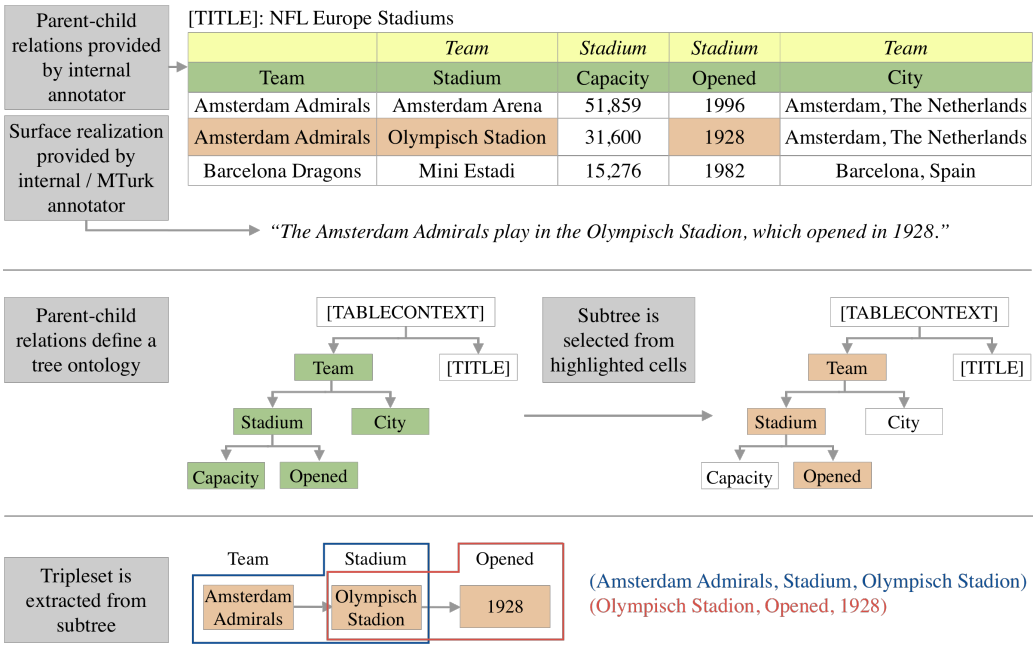

Figure 2: Overview of our human annotation procedure. *Top panel:* We collect the parent-child relations between columns from internal annotators (yellow is parent, green is child). Then, we collect a surface realization of the cells highlighted in orange. *Middle panel:* We use the provided parent-child relations to construct an ontology tree on the columns, then select the nodes corresponding to the highlighted cells. We gather a connected subtree by collecting all nodes leading up to the highlighted cells' lowest common ancestor. *Bottom panel:* We extract a set of triples from the subtree as shown. This triple-set is paired with the provided realization to form a DART instance.

were asked to write a description of the highlighted cells. We encouraged the annotators to use diverse vocabulary and syntactic structures. To ensure quality, internal annotators reviewed every crowd sourced sentence for correctness. They either rewrote or discarded the sentences that were nonsensical or incorrect. In some cases, they also changed cell highlighting patterns to match the sentence provided.

Build Tripleset-Sentence Pairs Finally, we convert the highlighted cells to triplesets. For a row R, we start with the table's column ontology T. We first place the cell values in R in their corresponding slots in T, e.g. in Figure 2 we fill TEAM with "Amsterdam Admirals". We then check that the nodes of T corresponding to the highlighted cells in R form a connected subtree. If not, we walk up the tree and highlight each traversed node up until the lowest common ancestor of the highlighted nodes (inclusive) to form a connected subtree. For each node N in the tree except the root node, we can extract the triple $(\text{parent}\,(N), \text{title}\,(N), N)$. For example, since STADIUM is highlighted in Figure 2, we extract the triple (Amsterdam Admirals, STADIUM, Olympisch Stadion). A small number of triple-sets contained more than 10 triples. We discarded these because their associated surface realizations were of poor quality. The numbers of tripleset-sentence pairs annotated by different annotators are shown in Table 2.

2.2 Automatically Converting Questions to Declarative Sentences

High quality natural language questions in open domain semantic parsing datasets such as WikiSQL and QA2D techniques found in automatically constructing NLI datasets (Demszky et al., 2018) present themselves as an attractive opportunity to semi-automatically construct an abundance of declarative sentences and align to table cells. We leveraged rule-based QA2D technique[2] together with manual screening to combine WikiSQL questions and SQL-retrieved-answers into declarative sentences and manually filtered out bad sentences.

We only execute SQL queries without aggregate commands[3] to retrieve answers corresponding to questions answerable by single rows. An example of such conversion is as follows:

[2]We use the rule-based model from https://github.com/kelvinguu/qanli (Demszky et al., 2018). The neural model code is not released.

[3]MAX, MIN, COUNT, SUM, AVG, JOIN, INTERSECT, UNION, GROUP BY, ORDER BY.

Question: *In which year did Greece hold its last Summer Olympics?*
Answer: *2004*
Declarative Sentence: *Greece held its last Summer Olympics in 2004.*

Alignment with table cells is done at two stages. We first align sentences with corresponding rows by changing SQL commands to SELECT * and use string matching to obtain columns and column headers relevant to the answer and WHERE condition. After manually filtering out bad sentences, bad alignments, or tables without ontology annotations, we were able to get 4,204 sentences. Finally, the corresponding table cells are then converted into triples in the same way as we described in Section 2.1.

Examples of produced declarative sentences can be found in Figure 10 of the Appendix.

2.3 Incorporating Existing Datasets

Since they provide a large amount of strictly aligned data-text pairs with high quality sentences, we incorporate the following existing datasets in the same ⟨triple-set, sentence⟩ pair format with some modifications.

WebNLG 2017 An instance of the WebNLG dataset contains a set of triples extracted from DBpedia and the target text written by human. We include the WebNLG 2017 dataset[4] consisting of 27731 triple-set sentence pairs with up to 7 RDF triples in a triple set covering 15 domains.

Cleaned E2E The original E2E dataset includes dialogue act meaning representations (MR) and natural language references in the restaurant domain. Later, Dušek et al. (2019) provide Cleaned E2E[5] by automatically fixing the dialogue acts to account for omissions and hallucinations in the text. We incorporate Cleaned E2E because of its strict alignment between the meaning representation and the text. To convert the MR to a triple-set, we take the NAME slot (present in almost all the MRs) as the subject. For example, the MR (NAME[ALIMENTUM], AREA[CITY CENTRE], FAMILYFRIENDLY[NO]) is converted to the triple-set {(ALIMENTUM, AREA, CITY CENTRE), (ALIMENTUM, FAMILYFRIENDLY, NO)}. We drop MRs which do not contain the NAME slot.

2.4 Predicate Unification

We canonicalized the predicates in our triple sets such that those of the same meaning are also represented the same. We manually constructed a predicate mapping table to achieve this. As an example, our predicate mapping maps "Hometown," "Home Town," and "Home Town/City" to the unified predicate "HOMETOWN."

After unifying predicates, we evaluated the diversity of DART by counting the number of unique predicates in its partitions. As shown in Table 2, we see that the Wikipedia partition of DART contains much more unique predicates than the WebNLG and Cleaned E2E partitions combined, despite having smaller number of ⟨triple-set, sentence⟩ pairs. This contributes significantly to the domain diversity of DART. In addition, we can see that DART exhibits a great deal of topical variety in terms of number of unique triples and vocabulary size.

2.5 Dataset Split

For WebNLG 2017 and Cleaned E2E, we use their original data splits. For our annotation on WikiTableQuestions and WikiSQL, random splitting will make train, dev, and test splits contain similar tables and similar ⟨triple-set, sentence⟩ examples. Therefore, to increase the generalization challenge, we compare the table title and the table header to find similar tables, and make sure the model is evaluated on test split tables that are least similar to those used for training. We first sample some tables as a seed test set, and then compute Jaccard similarity[6] with remaining tables based on the titles and the headers. If a table has a Jaccard similarity greater than 0.5 with any of the tables in the test set, we add it into the test set. A similar process is repeated to create the dev set, and the remaining tables form the training set. This results in 62,659/6,980/12,552 sentences in the train/dev/test sets, respectively.

3 Experimental Results

We conduct experiments on DART and the WebNLG 2017 dataset, with an ablation study on

[4] https://gitlab.com/shimorina/webnlg-dataset/-/tree/master/webnlg_challenge_2017

[5] https://github.com/tuetschek/e2e-cleaning

[6] https://en.wikipedia.org/wiki/Jaccard_index

WebNLG to show the benefits of using DART for data augmentation.

3.1 Models

We investigate several state-of-the-art Data-to-Text generation models. We report results of the following models on DART-testset: (1) Bidirectional-LSTM with attention, for which we use 2-layer bi-LSTM for encoder, with 300 dimensional word embeddings (without using pretrained word vectors), 512 hidden units and 0.3 dropout rate for the decoder. (2) Transformer (Vaswani et al., 2017), previously used by Castro Ferreira et al. (2019) on the WebNLG dataset. The input is formed by linearizing the unordered triple set. (3) BART (Lewis et al., 2020), for which we report results of both BART-base and BART-large. (4) T5 (Raffel et al., 2020): we add the same prefix "translate Graph to English:" to the input, as it is used in Ribeiro et al. (2020). We report results of T5-small, T5-base and T5-large models. For both BART and T5 models, we use implementations of Ribeiro et al. (2020), with same hyperparameter setting.

3.2 Evaluation Metrics

We use a variety of automatic metrics and human evaluation (Section 4) to evaluate the quality of the generated text. We report BLEU, METEOR, and TER which are used in the official WebNLG challenge. However, these measures have limitations in considering the semantic meanings of words or phrases (Novikova et al., 2017a), therefore we also report MoverScore (Zhao et al., 2019), BERTScore (Zhang et al., 2020), and BLEURT (Sellam et al., 2020) that incorporate semantics rather than surface forms using contextual embeddings. Furthermore, we include PARENT (Dhingra et al., 2019) which explicitly aligns n-grams from the reference and generated text to the data contents.

3.3 Results

DART Our experimental results on DART are summarized in Table 3. The T5-large model has the highest performance among all models with a BLEU score of 50.66. We attribute this to T5's generalization and transfer learning ability due to pretraining on multi-tasks. We can see that in general, pretrained models outperform others by a large margin, and increasing the model size seems to further boost the performance on DART. However, language models such as BART and T5 are pretrained by reconstructing text and, as a result, we found that their output on DART often contains hallucinated words (Parikh et al., 2020; Harkous et al., 2020; Reiter, 2020), as shown in Figure 11. In addition, while the pretrained model shows better text generation quality due to its generalization ability from pretraining, it does not fully capture the hierarchical ontology nature of the triple sets in their linearized input, therefore making DART more challenging. We suspect that models that are better at exploiting the ontology structure preserved in the input tripleset will achieve better performance on DART.

WebNLG Furthermore, we investigate if DART can improve pretrained models' performance on other Data-to-Text generation tasks. To this end, we finetune the baseline transformer model, BART-[base, large] and T5-[small, base, large] on the WebNLG 2017 dataset, and augment the training by adding instances in the DART training set. The experimental results can be found in Table 4. We report performances of some competitive models that are not pretrained, as well as the state-of-the-art performances of pretrained models on the WebNLG 2017 dataset by Ribeiro et al. (2020). On the bottom panel, we include results of experiments augmented with DART instances whose triplesets are generated with table ontology annotation, paired with human written sentences. We are able to achieve new state-of-the-art results on all WebNLG 2017 test set splits (seen, unseen and all) by finetuning T5-large on DART. We observe that using DART for data augmentation consistently improves the performance across all models, including the baseline transformer model that is not pretrained. Furthermore, we observe that more improvement is shown on unseen split of the test set, due to DART's open-domain nature. See Figure 12 of the Appendix for example model outputs aligned with their human references.

3.4 Ablation Study

We also conduct an ablation study on the WebNLG dataset to investigate what part of DART contributes most to improving the Data-to-Text tasks in general. We report results of the study in Table 6 of the Appendix. We divide DART into 4 partitions, where declarative sentence (auto-generated) partition and human annotated sentence partition contain instances whose triplesets are extracted from Wikipedia tables based on ontology. E2E partition contains instances converted from the E2E

	BLEU ↑	METEOR ↑	TER ↓	MoverScore ↑	BERTScore(F1) ↑	BLEURT ↑	PARENT ↑
LSTM with Attention	29.66	0.27	0.63	0.31	0.90	-0.13	0.35
End-to-End Transformer	27.24	0.25	0.65	0.25	0.89	-0.29	0.28
BART-base	47.11	0.38	0.46	0.51	0.95	0.37	0.55
BART-large	48.56	0.39	0.45	0.52	0.95	0.41	0.57
T5-small	47.69	0.39	0.46	0.52	0.95	0.40	0.56
T5-base	49.21	0.40	0.44	0.53	0.95	0.43	0.57
T5-large	**50.66**	**0.40**	**0.43**	**0.54**	**0.95**	**0.44**	**0.58**

Table 3: Model results on the test set of DART ↑: Higher is better. ↓: Lower is better.

	BLEU ↑			METEOR ↑			TER ↓		
	SEEN	UNSEEN	ALL	SEEN	UNSEEN	ALL	SEEN	UNSEEN	ALL
Pipeline Transformer† (Castro Ferreira et al., 2019)	56.28	23.04	42.41	0.42	0.21	0.32	0.39	0.63	0.50
Pipeline GRU† (Castro Ferreira et al., 2019)	56.09	25.12	42.73	0.42	0.22	0.33	0.39	0.64	0.51
MELBOURNE (Gardent et al., 2017)	54.52	33.27	45.13	0.41	0.33	0.37	0.40	0.55	0.47
BestPlan † (Moryossef et al., 2019)	53.30	34.41	47.24	0.44	0.34	0.39	0.47	0.56	0.51
DualEnc (Zhao et al., 2020)	63.45	36.73	51.42	0.46	0.37	0.41	0.34	0.55	0.44
PlanEnc (Zhao et al., 2020)	64.42	38.23	52.78	0.45	0.37	0.41	0.33	0.53	0.42
Ribeiro et al. (2020)									
BART-base ‡	63.02	41.74	53.36	0.45	0.35	0.40	0.33	0.52	0.42
BART-large ‡	63.71	44.17	54.95	0.46	0.39	0.42	0.33	0.51	0.41
T5-small ‡	65.30	45.58	56.57	0.46	0.39	0.43	0.32	0.49	0.40
T5-base ‡	64.89	52.86	59.44	0.46	0.42	0.44	0.33	0.42	0.37
T5-large ‡	64.89	54.01	59.95	0.46	0.43	0.44	0.34	0.41	0.37
+ DART									
BART-base	62.36	46.21	55.14	0.44	0.37	0.41	0.34	0.45	0.39
BART-large	64.51	50.20	58.06	0.46	0.40	0.43	0.32	0.44	0.38
T5-small	65.05	47.81	57.32	0.46	0.40	0.43	0.33	0.46	0.39
T5-base	65.42	50.71	58.80	0.46	0.41	0.44	0.32	0.43	0.37
T5-large	**65.82**	**56.01**	**61.44**	**0.46**	**0.43**	**0.45**	**0.32**	**0.38**	**0.35**

Table 4: The WebNLG 2017 results on the test set. †: We report results from Zhao et al. (2020) who use the evaluation scripts that are strictly the same as the official challenge.‡: We report results calculated with the model outputs on the WebNLG 2017 testset released by Ribeiro et al. (2020).

Tripleset source	**Sentence source**	**% fluent**	**% faithful**	**% (fluent+ mostly fluent)**	**% (faithful+ mostly faithful)**
WikiTableQuestions (§ 2.1)	human-written reference	75%	81%	96%	99%
	BART-base	74%	57%	93%	84%
	T5-base	72%	54%	94%	76%
WikiSQL (§ 2.2)	auto-generated reference	59%	56%	87%	88%
	BART-base	66%	51%	92%	83%
	T5-base	75%	65%	97%	90%

Table 5: Human evaluation over references and model outputs.

dataset, and WebNLG partition keeps the original data format. In general, we observe that adding DART instances that contain human written sentences brings most improvement, especially on unseen split. While adding E2E partition boosts the scores on seen test split and deteriorates the performance on unseen test split. This trend is consistent across all models. Comparing results of declarative sentence partition and human written sentence partition, we see that for most of the models, DART instances with human written sentences have better quality as it brings more improvement to the task.

4 Human Evaluation

In Table 5, we perform human evaluation on DART based on two criteria: (1) *fluency* if a sentence is natural and grammatical, and (2) semantic *faithfulness* if a sentence is supported by the input triples. We defined three levels of fluency: fluent, mostly fluent, and not fluent, and the same for semantic faithfulness. We ask 5 internal annotators to evaluate on 100 triplesets sampled from declarative sentence partition and another 100 triplesets sampled from human written sentence partition. Each

tripleset is paired with 3 sentences, one of them is the reference sentence, and the other two are outputs of BART-base and T5-base models.

The results in Table 5 attest to the high quality of our annotations since the human written references achieve highest fluency and faithfulness comparing to outputs of two strong baseline models. The evaluation on faithfulness also demonstrates that there is a considerable gap between the DART reference and the outputs of the state-of-the-art pretrained model, showing that there is a large room for improvement. We also noticed that the auto-generated declarative sentences are not as fluent or faithful as the model outputs because they are generated with a rule-based system. However, we decided to release this partition, along with other partitions of DART because it demonstrates an economic way to obtain large amounts of DART instances and it also shows benefits for generalization due to the diverse topics it contains.

5 Related Work

Data-to-Text Data-to-Text generation aims to produce natural language output from structured input. Applications include generating sports commentaries (Chen and Mooney, 2008; Wiseman et al., 2017), weather forecasts (Liang et al., 2009; Konstas and Lapata, 2012), biographical texts (Lebret et al., 2016; Liu et al., 2018), knowledge-base descriptions (Gardent et al., 2017), dialogue response generation (Wen et al., 2015, 2016), and commonsense reasoning (Lin et al., 2020). Yet, most existing datasets are restricted to specific domains and applications. In contrast, a major source of DART is from Wikipedia tables covering various domains and topics.

Representation of Data The input of the Data-to-Text datasets take different formats, including slot-value pairs, Abstract Meaning Representation (AMR) (Song et al., 2017; Ribeiro et al., 2019), Minimal Recursion Semantics (MRS) (Hajdik et al., 2019), Resource Description Framework (RDF triples) (Gardent et al., 2017), and logic forms (Chen et al., 2020b). There are also studies of converting tabular data to RDF triples in the Semantic Web community (Kellogg et al., 2015). Recently, some open-domain table-to-text datasets have been proposed including WikiTableText (Bao et al., 2018), LogicNLP (Chen et al., 2020a), and ToTTo (Parikh et al., 2020), whose inputs are rows or entire tables. In ToTTo, highlighted cells are also provided as input, and the authors found using only highlighted cells with flat row and column headers led to higher performance than using the entire table.

In contrast, DART is constructed by first annotating the tree-structured table ontology that encodes the semantic dependencies among table headers, and we could flexibly incorporate additional contexts such as the table title to the ontology tree. We then use an automatic procedure to extract connected components from the tree to form the input of a DART instance. Our annotation framework not only provides a flexible way of incorporating any contexts to the representation of tables, but also encodes hierarchical relationships among table headers and contexts, ensuring the extracted triples are logically consistent and can be described in text without loss of information.

Model Traditional Data-to-Text models break the generation progress into different stages such as signal analysis, data interpretation, document planning, microplanning, and realization (Reiter and Dale, 2000; Reiter, 2007). Recently, neural encoder-decoder models based on attention and copy mechanisms have shown promising results (Gehrmann et al., 2018; Puduppully et al., 2018, 2019; Castro Ferreira et al., 2019). Furthermore, recent progress on pretrained models such as GPT-2 (Radford et al., 2018), BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) has shown effective results for text generation tasks on machine translation, summarization, and conversation response generation. Chen et al. (2020c); Peng et al. (2020); Kale (2020) also finetune pretrained models on Data-to-Text tasks.

6 Conclusion

In this paper, we introduce DART, an open-domain corpus for structured data record to text generation. DART's ontology-preserving representation of data inputs differentiates itself from other open-domain Data-to-Text corpora. We found that DART introduces new challenges to several state-of-the-art Data-to-Text models due to its open-domain nature and its ontology structure of the semantic triple input. Furthermore, we found that using it for data augmentation improves other Data-to-Text tasks. For future work, we will explore more controlled, high-fidelity generation that better incorporates the ontology hierarchy of data.

7 Ethics Statement

Our dataset is constructed by accumulating and processing resources from various existing datasets that are open to the public. In addition, we collect annotations on structure of tabular data and human written sentences that describe data records.

The existing resources that we utilize mainly consist of (1) tabular data from Wikipedia, (2) information of restaurants presented with dialogue-act meaning representation and its textual description (E2E), and (3) information of various entities and their relationship that are in 15 different categories of DBPedia, which is a knowledge base built on contents created in various Wikimedia projects (WebNLG). It is possible that there are biases in these resources, either in the tabular data or the textual description written by humans.

For additional annotations we collected, we have two groups of annotators participating: internal annotators who are the authors of this work, and external annotators recruited from the Amazon Mechanical Turk platform. On MTurk, we use a pay rate of $15 per hour approximately based on our estimation of the time it takes to complete our annotation tasks. In total, it took 125 hours to complete all tasks on the Amazon Mechanical Turk platform. There are three annotation tasks: (1) Annotators are asked to specify ontological structure of the table by indicating relationship between table column headers, (2) Annotators are asked to write descriptions that are fluent and semantically faithful to the data records presented to them, and (3) Annotators are asked to evaluate sentences that are either references or model generated outputs. We acknowledge that it is also possible to have biases in the sentences written by the annotators, or in the data records that are presented to them.

We conducted experiments on our own dataset and the WebNLG dataset using BART and T5, two large-scale pretrained models. Both models are trained on large amounts of textual data such as news, books, and web text, which may contain any kinds of biases. As a result, it is possible to insert those biases into the models.

In total, we conducted 43 experiments: 7 on DART and 36 for our ablation study on the WebNLG dataset. We use a single NVIDIA V100 GPU for all experiments and each experiment took from 5 to 40 hours depending on the model size.

Acknowledgement

The authors would like to thank the anonymous reviewers for their discussion and feedback.

References

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *AAAI*.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *EMNLP*.

Alison J Cawsey, Bonnie L Webber, and Ray B Jones. 1997. Natural language generation in health care.

David L Chen and Raymond J Mooney. 2008. Learning to sportscast: a test of grounded language acquisition. In *ICML*.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020a. Logical natural language generation from open-domain tables. In *ACL*.

Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. 2020b. Logic2Text: High-fidelity natural language generation from logical forms. In *Findings of EMNLP*.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2020c. Few-shot nlg with pre-trained language model. In *ACL*.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *arXiv preprint arXiv:1809.02922*.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *ACL*.

Ondřej Dušek, David M. Howcroft, and Verena Rieser. 2019. Semantic noise matters for neural natural language generation. In *INLG*.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-REx: A large scale alignment of natural language with knowledge base triples. In *LREC*.

Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. 2019. Using local knowledge graph construction to scale Seq2Seq models to multi-document inputs. In *EMNLP-IJCNLP*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *INLG*.

Sebastian Gehrmann, Falcon Dai, Henry Elder, and Alexander Rush. 2018. End-to-end content and plan selection for data-to-text generation. In *INLG*.

Valerie Hajdik, Jan Buys, Michael Wayne Goodman, and Emily M Bender. 2019. Neural text generation from rich semantic representations. In *NAACL*.

Hamza Harkous, Isabel Groves, and Amir Saffari. 2020. Have your text and use it too! end-to-end neural data-to-text generation with semantic fidelity. *arXiv preprint arXiv:2004.06577*.

Mihir Kale. 2020. Text-to-text pre-training for data-to-text tasks. *arXiv preprint arXiv:2005.10433*.

Gregg Kellogg, Ivan Herman, and Jeremy Tandy. 2015. Generating RDF from tabular data on the web. W3C recommendation, W3C. Https://www.w3.org/TR/2015/REC-csv2rdf-20151217/.

Ioannis Konstas and Mirella Lapata. 2012. Unsupervised concept-to-text generation with hypergraphs. In *NAACL*.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *EMNLP*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. In *ACL*.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of EMNLP*.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *AAAI*.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019. Step-by-step: Separating planning from realization in neural data-to-text generation. In *NAACL*.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017a. Why we need new evaluation metrics for nlg. In *EMNLP*.

Jekaterina Novikova, Ondrej Dusek, and Verena Rieser. 2017b. The E2E dataset: New challenges for end-to-end generation. In *SIGDIAL*.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *EMNLP*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *ACL*.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog. In *arXiv*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2018. Data-to-text generation with content selection and planning. In *AAAI*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with entity modeling. In *ACL*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report, OpenAI.

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

Ehud Reiter. 2007. An architecture for data-to-text systems. In *Proceedings of the Eleventh European Workshop on Natural Language Generation (ENLG 07)*.

Ehud Reiter. 2020. Openai gpt system: What does it do? Technical report, Arria.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing AMR-to-text generation with dual graph representations. In *EMNLP*.

Leonardo F. R. Ribeiro, Martin Schmitt, Hinrich Schütze, and Iryna Gurevych. 2020. Investigating pretrained language models for graph-to-text generation. *arXiv*.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *ACL*.

Linfeng Song, Xiaochang Peng, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2017. Amr-to-text generation with synchronous node replacement grammar. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Pavlos Vougiouklis, Hady ElSahar, Lucie-Aimée Kaffee, Christophe Gravier, Frédérique Laforest, Jonathon S. Hare, and Elena Simperl. 2018. Neural wikipedian: Generating textual summaries from knowledge base triples. *Journal of Web Semantics*, 52-53:1 – 15.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, David Vandyke, and Steve Young. 2016. Conditional generation and snapshot learning in neural dialogue systems. In *EMNLP*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *EMNLP*.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *EMNLP*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. In *ICLR*.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *ACL*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *EMNLP*.

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

Appendix

The Appendix contains the following contents:

- Results of the ablation study on WebNLG 2017 testset.
- Statistics of the table ontology annotations.
- Examples of tables that help illustrate DART's annotation procedure.
- Examples of model outputs.

Model	Experiment	BLEU ↑			METEOR ↑			TER ↓		
		SEEN	UNSEEN	ALL	SEEN	UNSEEN	ALL	SEEN	UNSEEN	ALL
Baseline Transformer	[1] webnlg	49.81	5.51	31.81	0.39	0.09	0.24	0.47	0.86	0.64
	[2] webnlg+dart_decl_sents	52.31	**8.96**	**39.98**	0.40	0.07	0.25	0.45	0.79	0.60
	[3] webnlg+dart_human_annotated	53.68	7.02	36.36	0.40	**0.09**	**0.26**	0.43	**0.79**	**0.59**
	[4] webnlg+dart_ontology	53.40	8.54	38.51	**0.41**	0.08	0.26	0.44	0.80	0.60
	[5] webnlg+dart_e2e	51.76	5.92	32.36	0.40	0.09	0.25	0.45	0.86	0.63
	[6] webnlg+dart_full	**54.99**	8.64	39.11	0.40	0.08	0.25	**0.42**	0.81	0.60
BART-base	[1] webnlg	63.02	41.74	53.36	0.45	0.35	0.40	0.33	0.52	0.42
	[2] webnlg+dart_decl_sents	62.71	42.51	53.64	0.45	0.36	0.40	0.34	0.51	0.41
	[3] webnlg+dart_human_annotated	62.36	46.21	55.14	0.44	0.37	0.41	0.34	0.45	0.39
	[4] webnlg+dart_ontology	62.62	**46.74**	**55.54**	0.44	**0.38**	**0.41**	0.34	**0.45**	**0.39**
	[5] webnlg+dart_e2e	**64.00**	35.07	51.17	**0.45**	0.33	0.40	**0.33**	0.61	0.46
	[6] webnlg+dart_full	63.66	45.48	55.52	0.45	0.37	0.41	0.33	0.47	0.40
BART-large	[1] webnlg	63.71	44.17	54.95	0.46	0.39	0.42	0.33	0.51	0.41
	[2] webnlg+dart_decl_sents	65.18	46.79	56.79	0.46	0.39	0.42	0.32	0.48	0.40
	[3] webnlg+dart_human_annotated	64.51	**50.20**	**58.06**	0.46	**0.40**	**0.43**	0.32	**0.44**	**0.38**
	[4] webnlg+dart_ontology	64.19	49.62	57.65	0.46	0.39	0.43	0.33	0.45	0.38
	[5] webnlg+dart_e2e	65.06	30.17	48.24	0.46	0.33	0.40	0.32	0.69	0.49
	[6] webnlg+dart_full	**65.24**	47.96	57.44	**0.46**	0.39	0.43	**0.32**	0.46	0.39
T5-small	[1] webnlg	65.30	45.58	56.57	0.46	0.39	0.43	0.32	0.49	0.40
	[2] webnlg+dart_decl_sents	64.18	46.61	56.27	0.46	0.39	0.43	0.33	0.48	0.40
	[3] webnlg+dart_human_annotated	65.05	**47.81**	**57.32**	0.46	**0.40**	**0.43**	0.33	**0.46**	**0.39**
	[4] webnlg+dart_ontology	65.17	47.49	57.24	0.46	0.39	0.43	0.32	0.47	0.39
	[5] webnlg+dart_e2e	**65.56**	41.28	54.56	**0.46**	0.38	0.42	**0.32**	0.54	0.42
	[6] webnlg+dart_full	64.70	47.56	57.01	0.46	0.39	0.43	0.33	0.47	0.39
T5-base	[1] webnlg	64.89	**52.86**	59.44	0.46	**0.42**	0.44	0.33	**0.42**	0.37
	[2] webnlg+dart_decl_sents	65.44	50.80	58.81	0.46	0.41	0.44	0.32	0.43	0.37
	[3] webnlg+dart_human_annotated	65.42	50.71	58.80	0.46	0.41	0.44	0.32	0.43	0.37
	[4] webnlg+dart_ontology	65.17	51.49	59.04	0.46	0.41	0.44	0.33	0.43	0.37
	[5] webnlg+dart_e2e	65.11	49.64	58.19	0.46	0.41	0.44	0.33	0.46	0.39
	[6] webnlg+dart_full	**65.99**	51.68	**59.50**	**0.46**	0.42	**0.44**	**0.32**	0.43	**0.37**
T5-large	[1] webnlg	64.89	54.01	59.95	0.46	0.43	0.44	0.34	0.41	0.37
	[2] webnlg+dart_decl_sents	65.97	53.00	60.12	0.46	0.42	0.44	0.32	0.41	0.36
	[3] webnlg+dart_human_annotated	65.82	**56.01**	**61.44**	0.46	**0.43**	**0.45**	0.32	**0.38**	**0.35**
	[4] webnlg+dart_ontology	65.53	55.20	60.90	0.46	0.42	0.44	0.32	0.38	0.35
	[5] webnlg+dart_e2e	**66.27**	54.13	60.76	**0.46**	0.43	0.45	**0.32**	0.41	0.36
	[6] webnlg+dart_full	65.78	54.35	60.64	0.46	0.42	0.44	0.32	0.39	0.35

Table 6: Results of ablation study on WebNLG 2017 testset. **dart_decl_sents** refers to DART partition that contains auto-generated declarative sentences mentioned in Section 2.2, **dart_human_annotated** refers to partition that contains human written sentences mentioned in Section 2.1, **dart_ontology** is the combination of **dart_decl_sents** and **dart_human_annotated**, and **dart_e2e** refers to DART partition containing instances extracted from E2E dataset, the process of which is mentioned in Section 2.3. Note that **dart_full** is the combination of **dart_ontology** and **dart_e2e**.

	Tables	Ontology depth (min, med, max)	Nodes in ontology (min, med, max)	Branching factor (mean)
WikiTableQuestions	2060	1, 1, 4	2, 6, 25	4.0
WikiSQL	3563	1, 1, 4	3, 7, 25	5.1

Table 7: Properties of the ontology in the WikiTableQuestions and WikiSQL samples in DART. Branching factor refers to the average number of children across all non-leaf nodes in a table's ontology.

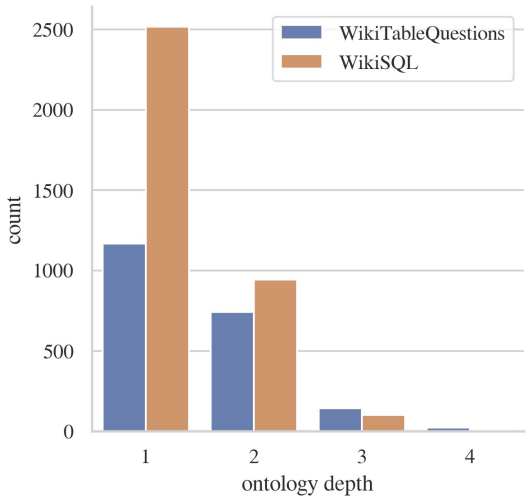

Figure 3: Distribution of column ontology depths in the WikiTableQuestions and WikiSQL samples in DART v1.1.1.

```
<entry category="MISC" eid="Id5" size="3">
  <modifiedtripleset>
    <mtriple>Apertura 2006 | JORNADA_OR_OTHER | Semifinals Ida</mtriple>
    <mtriple>Semifinals Ida | AWAY_TEAM | América</mtriple>
    <mtriple>Semifinals Ida | HOME_TEAM | Chivas</mtriple>
  </modifiedtripleset>
  <lex comment="WikiTableQuestions" lid="Id1">
      Chivas and América will compete in the semifinals of the Apertura 2006 tournament.
  </lex>
</entry>

<entry category="MISC" eid="Id76" size="6">
  <modifiedtripleset>
    <mtriple>Terry Jenkins | ROUND | 1st Round</mtriple>
    <mtriple>Terry Jenkins | YEAR | 2014</mtriple>
    <mtriple>[TABLECONTEXT] | [TITLE] | PDC World Darts Championship</mtriple>
    <mtriple>1st Round | OPPONENT | Per Laursen</mtriple>
    <mtriple>1st Round | RESULT | Lost</mtriple>
    <mtriple>[TABLECONTEXT] | PLAYER | Terry Jenkins</mtriple>
  </modifiedtripleset>
  <lex comment="WikiTableQuestions" lid="Id1">
      Terry Jenkins lost the game with Per Laursen in
      the 1st Round of 2014 PDC World Darts Championship
  </lex>
</entry>
```

Figure 4: Examples of DART instance

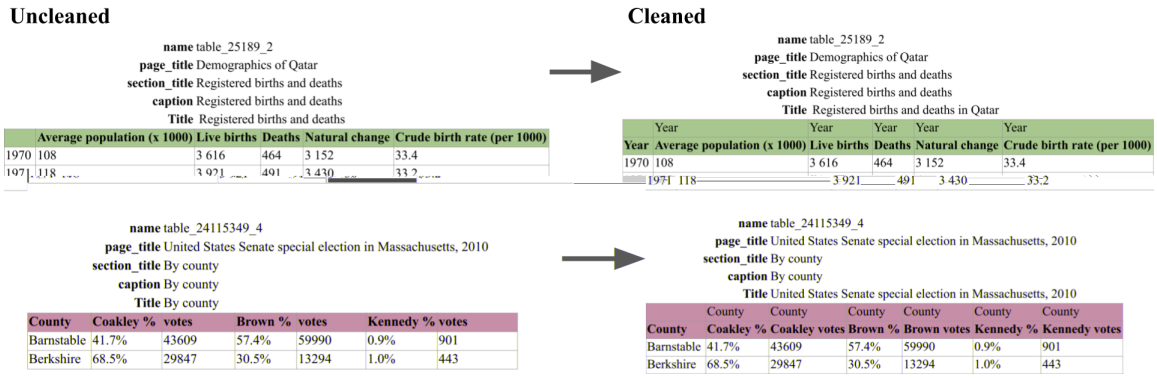

Figure 5: An example of the data cleaning. The top left table had a missing column name and the table title was not specific to the data; our internal annotators add the missing column name "Year" and linked the rest of the columns to the "Year" column. The bottom left table had repeat column names in the table; our internal annotators disambiguate the columns by making the column names more specific.

Table Title: Casper Elgaard

[TITLE]	[TITLE]	[TITLE]	[TITLE]	[TITLE]	[TITLE]	[TITLE]	Class
Year	Team	Co-Drivers	Car	Class	Laps	Pos.	Class Pos.
2001	Team Den Blå Avis / Team Goh	John Nielsen & Hiroki Katoh	Dome S101-Judd	LMP900	66	DNF	DNF
2003	RN Motorsport	John Nielsen & Hayanari Shimoda	DBA4 03S-Zytek	LMP675	288	22nd	2nd
2004	Lister Racing	John Nielsen & Jens Møller	Lister Storm LMP-Chevrolet	LMP1	279	24th	9th
2006	Zytek Engineering / Team Essex Invest	John Nielsen & Philip Andersen	Zytek 06S	LMP1	269	NC	NC
2007	Aston Martin Racing Larbre	Christophe Bouchut & Fabrizio Gollin	Aston Martin DBR9	GT1	341	7th	3rd
2008	Team Essex	John Nielsen &Sascha Maassen	Porsche RS Spyder Evo	LMP2	347	12th	2nd
2009	Team Essex	Kristian Poulsen & Emmanuel Collard	Porsche RS Spyder Evo	LMP2	357	10th	1st
2011	Hope Racing	Steve Zacchia & Jan Lammers	Oreca 01-Swiss HyTech	LMP1	115	DNF	DNF

Figure 6: A WikiTableQuestions table that uses [TITLE] in the ontology.

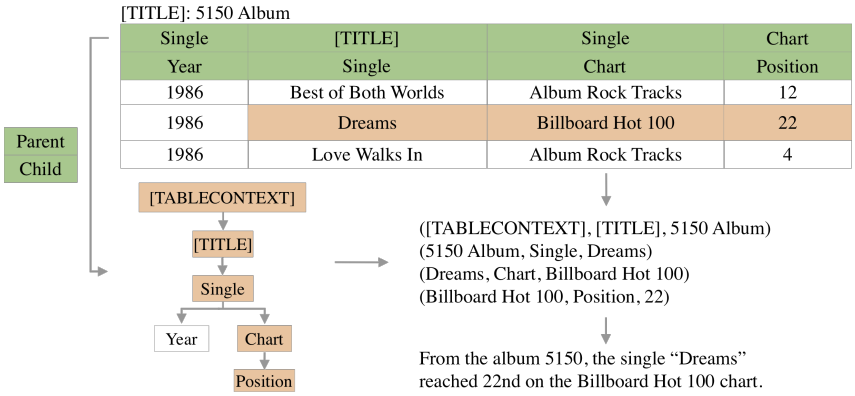

Figure 7: A manually annotated table from WikiTableQuestions with a sentence that uses the table title.

Table Title: Peter De Villiers

Competition	Event	Competition	[TITLE]	Position	[TITLE]	
Year	Competition	Venue	Position	Event	Time	
2000	World Junior Championships	Santiago, Chile	2nd	400 m hurdles	50.52	The competitor won a silver medal at the 2000 World Junior Championships in Santiago, Chile.
2000	World Junior Championships	Santiago, Chile	4th	4x400 m relay	3:07.66	
2005	World Championships	Helsinki, Finland	17th (sf)	400 m hurdles	49.75	
2005	World Championships	Helsinki, Finland	13th (h)	4x400 m relay	3:04.64	
2006	Commonwealth Games	Melbourne, Australia	7th	400 m hurdles	50.51	They finished the 400 m hurdles 7th in 50.51.
2006	African Championships	Bambous, Mauritius	4th	400 m hurdles	50.96	In Bambous, Peter De Villiers came in 4th in the African Championships.
2007	All-Africa Games	Algiers, Algeria	2nd	400 m hurdles	48.91	
2007	All-Africa Games	Algiers, Algeria	8th	4x400 m relay	DNF	The 2007 All-Africa Games held a 4x400 m relay.
2007	World Championships	Osaka, Japan	15th (sf)	400 m hurdles	49.37	
2008	Olympic Games	Beijing, China	12th (sf)	400 m hurdles	49.44	He competed in the 2008 Beijing Olympic Games.

Figure 8: A manually annotated table from WikiTableQuestions. Annotators created a table ontology, and they wrote sentences encapsulating the information in the orange cells for a given row. Whenever a sentence referenced the table title, that sentence was also highlighted green.

Title: 1950 Indianapolis 500

Pos	No	Driver	Constructor	Lap	Gap
7	69	Duke Dinsmore	Kurtis Kraft-Offenhauser	4:34.67	+ 6.70

Duke Dinsmore, number 69, finished in 7th position at the 1950 Indianapolis 500.

y

Title: Kurt Maschler Award

Year	Author	Illustrator	Title	Publisher
1991	Colin McNaughton	McNaughton	Have You Seen who's just moved in next door to us?	Walker

"Have You Seen who's just moved in next door to us?" was published by Walker in 1991.

n

Title: Maxi Priest

Year	Song	U.S.	U.S. R&B	U.S. AC	UK	Album
1990	“Just a Little Bit Longer"	62	30	None	62	Bonafide

Max Alfred "Maxi" Elliott known by his stage name Maxi Priest, is a British reggae vocalist of Jamaican descent.

x

Figure 9: An example of collected MTurk-generated sentences for WikiTableQuestions. Internal annotators went through the generated sentences and checked for both sentence coherence and title usage. Below the generated sentences, ‘y’ meant the sentence references the table title, ‘n’ meant the sentence did not use the table title, ‘x’ meant the sentence was nonsensical.

Title: Harrisburg City Islanders

Year	Division	League	Regular Season	Playoffs	U.S. Open Cup	Avg. Attendance
2012	3	usl pro	6th	quarterfinals	quarterfinals	1452

A team in the usl pro league has an average attendance of 1452.

n

Year	Division	League	Regular Season	Playoffs	U.S. Open Cup	Avg. Attendance
2009	3	usl second division	3rd	semifinals	quarterfinals	1857

In 2009, the Harrisburg City Islanders got semifinals in Playoffs in league usl second division.

y

Title: Cincinnati Kings

Year	Division	League	Regular Season	Playoffs	Open Cup
2008	4	usl pdl	4th, great lakes	did not qualify	did not qualify

When 4th, great lakes is the regular season usl pdl is the league.

x

Figure 10: Automatically generated declarative sentences from WikiSQL with human validation. Annotators went through the generated sentences and checked for both sentence coherence and title use. Below the generated sentences, ‘y’ meant the sentence references the table title, ‘n’ meant the sentence did not use the table title, ‘x’ meant the sentence was nonsensical.

```
- Sample 1 -
Input triples:
<H> Peru Earthquake <R> scale of disaster <T> 250k homeless
<H> Peru Earthquake <R> year <T> 2007

BART-base output: 250k people were killed in the 2007 philippine earthquake .

- Sample 2 -
Input triples:
<H> [TABLECONTEXT] <R> game <T> 3
<H> 3 <R> attendance <T> 10 637
<H> [TABLECONTEXT] <R> [title] <T> 2006 Minnesota Swarm season

BART-base output: the minnesota swarm played in front of a crowd of 10 , 684 people .

- Sample 3 -
Input triples:
<H> Andrew Phelps McCormick <R> state <T> TX
<H> Andrew Phelps McCormick <R> active <T> 1892-1916

T5-base output: andrew phelps mccormick was active from 1892 to 1616 in texas .
```

Figure 11: Examples of hallucinated outputs of pretrained models trained on DART

```
- Sample 1 -
Input triples:
    <H> Andrew Rayel <R> associated Band/associated Musical Artist <T> Christian Burns
    <H> Andrew Rayel <R> associated Band/associated Musical Artist <T> Jonathan Mendelsohn

reference:
andrew rayel , is associated with musical artist jonathan mendelsohn and christian burns .

train on WebNLG - BART-base output:
christian mendelsohn and andrew rayel are both associated with the same band , christian burns .

train on DART - BART-base output:
andrew rayel is associated with christian burns and jonathan mendelsohn .

- Sample 2 -
Input triples:
<H> Indie rock <R> stylistic Origin <T> New wave music

reference: the stylistic origin of indie rock is new wave music .

train on WebNLG - BART-base output:
the alternative rock genre is new wave .

train on DART - BART-base output:
indie rock is influenced by new wave music .

- Sample 3 -
Input triples:
<H> Abradab <R> associated Band/associated Musical Artist <T> Magik rapper
<H> Abradab <R> associated Band/associated Musical Artist <T> Kaliber 44

reference:
abradab , an artist for the band kaliber 44 , is associated with magik ( rapper ) .

train on WebNLG - BART-base output:
magiber 44 is the creator of abradab , which is also associated with the magik rapper .

train on DART - BART-base output:
magik rapper and kaliber 44 are the associated musicians of abradab .


- Sample 4 -
Input triples:
<H> Alfa Romeo 164 <R> assembly <T> Milan
<H> Alfa Romeo 164 <R> related Mean Of Transportation <T> Saab 9000

reference:
the alfa romeo 164 , which is assembled in milan , is a related means of transportation to saab 9000 ,
in that they are both cars .

train on WebNLG - T5-base output:
alfa romeo 164 is a transport vehicle for saab 9000 and is found in milan .

train on DART - T5-base output:
alfa romeo 164 ( assembled in milan ) is a related transport vehicle to saab 9000 .

- Sample 5 -
Input triples:
<H> Akeem Ayers <R> former Team <T> Tennessee Titans
<H> Akeem Ayers <R> draft Pick <T> 39

reference:
akeem ayers ' former team was tennessee titans and he was number 39 in the draft pick .

train on WebNLG - T5-large output:
akeem ayers was drafted with the 39th pick by the tennessee titans .

train on DART - T5-large output:
akeem ayers , a former player of the tennessee titans , was the 39th draft pick .
```

Figure 12: Examples of model outputs - with or without DART data augmentation