DART: Open-Domain Structured Data Record to Text Generation

Dragomir Radev$^{1,2}$ Rui Zhang$^1$ Amrit Rau$^1$ Abhinand Sivaprasad$^1$ Chiachun Hsieh$^3$
Nazneen Fatema Rajani$^2$ Xiangru Tang$^5$ Aadit Vyas$^1$ Neha Verma$^1$ Pranav Krishna$^4$
Yangxiaokang Liu$^1$ Nadia Irwanto$^1$ Jessica Pan$^1$ Faiaz Rahman$^1$ Ahmad Zaidi$^1$
Murori Mutuma$^1$ Yasin Tarabar$^1$ Ankit Gupta$^1$ Tao Yu$^1$ Yi Chern Tan$^1$
Xi Victoria Lin$^2$ Caiming Xiong$^2$ Richard Socher$^2$

$^1$Yale University $^2$Salesforce Research $^3$The University of Hong Kong $^4$MIT $^5$University of the Chinese Academy of Sciences
{dragomir.radev, r.zhang}@yale.edu, {nazneen.rajani}@salesforce.com

Abstract

We introduce DART, a large dataset for open-domain structured DA Data Record to Text generation. We consider the structured data record input as a set of RDF entity-relation triples, a format widely used for knowledge representation and semantics description. DART consists of 82,191 examples across different domains with each input being a semantic RDF triple set derived from data records in tables and the tree ontology of the schema, annotated with sentence descriptions that cover all facts in the triple set. This hierarchical, structured format with its open-domain nature differentiates DART from other existing table-to-text corpora. We conduct an analysis of DART on several state-of-the-art text generation models, showing that it introduces new and interesting challenges compared to existing datasets. Furthermore, we demonstrate that finetuning pretrained language models on DART facilitates out-of-domain generalization on the WebNLG 2017 dataset.

1 Introduction

Automatically generating textual descriptions from structured data inputs is crucial to improving the accessibility of knowledge bases to lay users. Such applications include explaining data records to non-experts in the healthcare domain (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 research progress has been made in this field, there are still several issues with existing data-to-text datasets. First, they mostly deal with knowledge sources with a flat ontology structure such as slot-value pairs (e.g., WikiBio (Lebret et al., 2016), The E2E Dataset (Novikova et al., 2017b), dialog response generation (Wen et al., 2016)) and tables (e.g., RotoWire (Wiseman et al., 2017), WikiTableText (Bao et al., 2018), LogicNLP (Chen et al., 2020a), ToTTo (Parikh et al., 2020)). The flat structure representation is not powerful enough to encode rich semantic relationships in the ontology, for example, transitive functional dependencies such as person - city - population. ToTTo(Parikh et al., 2020)$^2$ is a recent table-to-text dataset pairing a one-sentence description with a Wikipedia table and a set of highlighted table cells. They found using only highlighted cells with flat row and column headers led to higher performance than using the whole table. Second, some of them only focus on a small number of domains (e.g., WebNLG (Gardent et al., 2017b) on 15 categories from DBPedia, WikiBio on biographies, E2E on restaurants, RotoWire on basketball, MLB on baseball (Puduppully et al., 2019), WeatherGov (Liang et al., 2009) on weather forecasts, RoboCup (Chen and Mooney, 2008) on soccer reports). Furthermore, some of them only have loose alignments between data input and sentence due to the automatic generation procedure (e.g., RotoWire, Neural Wikipedian (Vougiouklis et al., 2018), and T-Rex (Elsahar et al., 2018)).

To address some of these issues and to encourage further research in natural language generation from semantic data, we introduce DART, a large and open-domain structured DA Data Record to Text generation corpus. DART provides high-quality sentence annotations with each input being a set of entity-relation triples in a tree structure. As shown in Figure 2, to construct DART, we combine reliable human annotations and an automatic conversion procedure from two existing open-domain

---

$^1$DART is available at https://github.com/Yale-LILY/dart.

$^2$This paper appeared on arxiv after we had started working on DART.
Question Answering datasets, while also incorporating two other existing corpora. More specifically, we use open-domain tables from Wikipedia and ask human annotators to construct a tree-structured ontology of the column headers. Then we automatically choose a subset of the columns by sampling a connected component from the ontology tree, ensuring that the sampled subset is semantically valid and continuous. We present the table row with the chosen cells highlighted and ask the annotators to describe the highlighted parts in natural language. We illustrate this annotation procedure in Figure 1. Furthermore, we introduce an automatic construction procedure by converting the text-to-SQL dataset WikiSQL into tripleset-sentence pairs. To this end, we execute the SQL command to get the answer of the question and then convert the question-answer pair into a declarative sentence using a set of expert rules with part-of-speech and dependency parsing information (Demszky et al., 2018), and then we use a string matching heuristic to find corresponding table cells to build tripleset-sentence pairs.

Evaluating several state-of-the-art table-to-text models on DART, we found that while these models can achieve high results on domain-specific datasets, they don’t perform as well on DART due to its open-domain nature and structured ontology representations.

Our contributions are as follows. (1) We present a large and open-domain corpus for structured data record to text generation with each input as a set of entity-relation triples based on the tree ontology of the table. This hierarchical structured format differentiates our corpus from other data sets including both domain-specific and open-domain table-to-text corpora. (2) We benchmark on several state-of-the-art table-to-text models showing that DART introduces new open-domain generalization challenges with the hierarchical structure of
the semantic triple inputs. (3) We also demonstrate that using the training set of DART for data augmentation improves the pretrained language models on WebNLG 2017 because DART helps out-of-domain generalization due to its open-domain nature.

2 Related Work

Data-to-text generation aims to produce natural language output from structured input. Applications include generating sports commentaries (Tanaka-Ishii et al., 1998; Chen and Mooney, 2008; Wiseman et al., 2017; Wang, 2019; Taniguchi et al., 2019), weather forecasts (Goldberg et al., 1994; Reiter et al., 2005; Liang et al., 2009; Konstas and Lapata, 2012; Mei et al., 2016), biographical texts (Lebret et al., 2016; Sha et al., 2018; Liu et al., 2018; Vougiouklis et al., 2018; Perez-Beltrachini and Lapata, 2018), knowledge-base descriptions (Oâ`/A ´Z-Donnell et al., 2000; Banik et al., 2013; Gardent et al., 2017a,b; Novikova et al., 2017b; Wang et al., 2018; Yu et al., 2019), code comment (Iyer et al., 2016), dialogue response generation (Wen et al., 2015, 2016), and commonsense reasoning (Lin et al., 2019; Rajani 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.

Data-to-text datasets take different formats, including slot-value pairs, Abstract Meaning Representation (AMR) (Flanigan et al., 2016; Ferreira et al., 2017; Song et al., 2017; Zhu et al., 2019a; Ribeiro et al., 2019; Damonte and Cohen, 2019; Wang et al., 2020a; Zhao et al., 2020b; Shen et al., 2020), Minimal Recursion Semantics (MRS) (Hajdik et al., 2019), Resource Description Framework (RDF triples) (Gardent et al., 2017b; Vougiouklis et al., 2018; Distiawan et al., 2018; Elsahar et al., 2018; Zhu et al., 2019b), logic forms (Chen et al., 2020b), and tables. 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). While DART is also constructed by annotating tables, we introduce a tree-structured ontology on the table headers and an automatic procedure to extract connected components from the ontology as highlighted table cells. This encodes hierarchical relationships among table headers while ensuring the highlighted part is logically consistent and can be described in text without loss of information.

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 (Wiseman et al., 2017; Liu et al., 2018; Gehrmann et al., 2018; Puduppally et al., 2018, 2019; Iso et al., 2019; Castro Ferreira et al., 2019; Shen et al., 2019; Roberti et al., 2019; Zhu et al., 2019b; Koncel-Kedziorski et al., 2019; Ye et al., 2020; Harkous et al., 2020; Wang et al., 2020b; Song et al., 2020; Shahidi et al., 2020). Furthermore, recent progress on pretrained language models such as GPT-2 (Radford et al., 2018), UniLM (Dong et al., 2019), CTRL (Keskar et al., 2019), BERT-to-BERT (Rothe et al., 2020), DialoGPT (Zhang et al., 2020b), T5 (Raffel et al., 2019), and BART (Lewis et al., 2020) have shown very effective results for text generation tasks on machine translation, summarization, conversation response generation, and abstractive QA. Chen et al. (2020c); Baolin Peng (2020); Kale (2020) also fine-tune pretrained language models on data-to-text tasks.

To evaluate the performance of different NLG methods, qualitative evaluation measures such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) and METEOR (Banerjee and Lavie, 2005), are widely used to replace costly human judgments. However, these measures have limitations in considering the semantic meanings of words or phrases, which is shown in Novikova et al. (2017a) that these measures fail to correlate well with human judgement. Recently, embedding based evaluation measures have been proposed to tackle this problem. Typical examples include BERTScore (Zhang et al., 2020a), YiSi-1 (Lo, 2019), WMD (Kusner et al., 2015), WMDO (Chow et al., 2019), MoverScore (Zhao et al., 2019), and BLEURT (Sellam et al., 2020). They first compute the semantic similarity between word representations, produced by word embedding models such as word2vec (Mikolov et al., 2013) or contextual embedding models such as BERT (Devlin et al., 2019). The final score is then given by different intrinsic metrics, such as generalized precision, recall, and F-score used in BERTScore, and
Table 1: DART compared with other open-domain table-to-text datasets. DART takes triple sets as input by incorporating the tree ontology, and its surface realizations tend to be longer with more than single sentence verbalization. These statistics are computed from DART v1.1.1. SR: Surface Realization.

| 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  |

Earth Mover distance used in WMD, WMDO, and MoverScore. Furthermore, Dhingra et al. (2019) propose PARENT which explicitly aligns n-grams from the reference and generated text to the table information.

3 DART Data Collection

We build DART from existing datasets that cover a variety of different domains while allowing us to build a tree ontology and form RDF triple set as our semantic representation. The data statistics are summarized in Table 1 and 2. Compared with WebNLG and Cleaned E2E in Table 2, DART exhibits more topical variety in terms of the number of unique predicates, the number of unique triples, and the vocabulary size. Compared to other open-domain table-to-text datasets in Table 1, DART takes triple sets as input by incorporating the tree ontology, and its surface realizations tend to be longer with more than single sentence verbalization.

As illustrated in Figure 2, DART is constructed using multiple complementary methods: (1) human annotation on open-domain Wikipedia tables from WikiTableQuestions (Pasupat and Liang, 2015) and WikiSQL (Zhong et al., 2017) (Section 3.1), (2) automatic conversion of questions in WikiSQL to declarative sentences (Section 3.2), and (3) incorporation of existing datasets including WebNLG 2017 (Gardent et al., 2017a,b; Shimorina and Gardent, 2018) and Cleaned E2E (Novikova et al., 2017b; Dušek et al., 2018, 2019) (Section 3.3).

We also explored automatic alignments between the knowledge base and text including Neural Wikipedian (Vougiouklis et al., 2018) and T-Rex (Elsahar et al., 2018). Although these datasets are large in size with natural sentences in a variety of domains, their automatic data construction procedures cannot guarantee good quality alignment: the sentence is often noisy with omitted or hallucinated information compared with the paired triple. So we ended up not including these in the current version of DART.

3.1 Tree Ontology and Sentence Annotation on Tables

Tables are sources of structured data and have been used for question answering (WikiTableQuestions), semantic parsing (WikiSQL), and table-to-text generation (ToTTo) tasks. Here, we aim to collect triple-sentence pairs from open-domain Wikipedia tables in the WikiTableQuestions and WikiSQL datasets. However, such tables have a flat structure, making them not directly usable for building (subject, predicate, object) triples, which are critical in capturing rich relationships in the data. We propose a two-stage annotation process for constructing tripleset-sentence pairs based on a tree-structured ontology of each table. First, internal skilled annotators denote the parent column for each column header. Then, a larger number of annotators provide a sentential description of an automatically-chosen subset of table cells in a row. To form a tripleset-sentence pair, the highlighted cells can be converted to a connected tripleset automatically according to the column ontology for the given table.

Tree Ontology Annotation For each column in a given table, our internal annotators labeled its ontological parent. In Figure 1, for example, the annotator would provide the sequence {{blank), TEAM, STADIUM, STADIUM, TEAM} — column TEAM has no parent, STADIUM has parent TEAM, CAPACITY has parent STADIUM, and so on. In many cases, the relationship between a parent column and its child column can loosely be conceptualized as a “has-a” relationship.

In some tables, the table title itself (rather than a column) serves as the root of the column on-
ontology (e.g., Figure 4). In these cases, the annotator assigns [TITLE] as the parent of the relevant columns. Note that by construction, annotators are prohibited from assigning parents to [TITLE]. Ontologies are rejected and corrections are requested if the provided ontology is disconnected or contains a cycle.

Following the procedure above, we now have a tree structure in which each node is either a column in the table or [TITLE]. The tree root is either [TITLE] or a column name. However, in many cases, even if the table title is not included as a node in the ontology, it provides important context for understanding the table’s rows and the annotated sentence (e.g., Figure 5). To handle this case, we add a dummy [TABLECONTEXT] node as the parent of the existing tree’s root node. This context node doesn’t represent a data entity, but instead unifies the table’s title into the ontology. If the former root node was [TITLE], then we still need the [TABLECONTEXT] node which will have a single child, [TITLE]. The reason why we need a [TABLECONTEXT] node here is because the [TITLE] node needs a parent during triple formation.

In this way, we generate a fully-connected tree of column names and the two special nodes, [TABLECONTEXT] and [TITLE]. Every tree contains both special nodes. These trees exhibit a great deal of structural variety, and relevant statistics are summarized in Table 3 and Figure 3.

Some tables are malformed and have duplicate or missing column names (e.g., Figure 6). In these cases, human annotators either changed or added appropriate column names in order to fit these constraints. Additionally, we require that the annotated ontology encodes a valid tree structure with each of the column names present. We verify that all headers appear as a node, nodes refer only to column headers, and that each of these tree structures contains no cycles. We introduce no new “abstract” nodes to encode relationships between column header nodes for the sake of consistency, scale, and robustness. Note that for many tables, the determination of an ontology is a subjective process with many “correct” answers – for exam-
Table 3: 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. These statistics were computed on DART v1.1.1.

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

Connected Component Extraction  After we label the ontology, we automatically choose a subset of cells in a table row. 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 formulate a coherent sentence. For example, in Figure 1, 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. In addition, the component will always include the root (unless the root node is a table’s title) as it usually denotes the subject of the row.

The extracted components have two controllable properties: size and shape. To create variation in size, we randomly sampled between two and five inclusive. The shape can be characterized 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 parent-child node pairs creates a deeper tree. We created a sliding scale between wide and deep trees using an expansion parameter, $p$. Component search will recursively search through the ontology’s nodes. At each stage, it recursively visits 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 selected cells for sentence annotation.

Sentence Annotation  We collect annotations from two sources: internal annotators in our group (Figure 7) and Amazon Mechanical Turk (MTurk) workers (Figure 9). We showed both annotator groups the table title and table row. When collecting internal annotations, we asked annotators to highlight a subset of cells from the row. When collecting annotations from MTurk, we pre-selected and highlighted cells according to the connected component extraction method described above. Both annotator groups were then asked to write natural sentences describing the highlighted cells.

We encouraged the annotators to use diverse vocabulary and syntactic structures. We also asked annotators whether or not they used information (either explicit or entailed) from the table title in producing each sentence. In particular, this information is used to determine whether or not the [TITLE] node should be included in the ontol-
ology subtree of interest. In our WikiSQL sentences, much of which came from automatic declarative sentence generation, 5.5% of sentences included the table title. In our WikiTableQuestions sentences which were internally annotated, 43.0% of sentences included the title.

A small-scale review revealed that a number of the sentences provided were nonsensical, ungrammatical, or incorrect. Additionally, we found that annotators often misreported whether or not they used information from table titles in producing sentences. Therefore, a team of skilled annotators reviewed every crowdsourced sentence and marked whether it (a) was correct and (b) used information from the table title or not. The skilled annotators 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 triples according to the ontology annotation as shown in Figure 1. To do this for a row $R$, we start with the source table’s column ontology $T$. To extract triples from $R$, we first place the cell values in $R$ in their corresponding slots in $T$. For the example ontology in Figure 1, we fill $TEAM$ with “Amsterdam Admirals,” $STADIUM$ with “Olympisch Stadion,” etc. We then check that the nodes of $T$ corresponding to the highlighted cells in $R$ form a connected subtree. If this is not the case, we walk up the tree, highlighting each traversed node up until the lowest common ancestor of the highlighted nodes (inclusive). The selected nodes form a connected subtree. For each node $N$ in the tree except the root node, we can extract the triple $(parent(N), title(N), N)$. For example, since $STADIUM$ is highlighted in Figure 1, we extract the triple (Amsterdam Admirals, Stadium, Olympic Stadium).

All the triples extracted from $R$ and $T$ together with their sentence annotation form a tripleset-sentence pair. A small number of triplesets contained more than 10 triples; we discarded these because their associated surface realizations were long, meandering, and of uniformly poor quality.

### 3.2 Automatically Converting Questions to Declarative Sentences

WikiSQL is a dataset of questions and corresponding SQL queries for tables from Wikipedia. We convert these questions to declarative sentences and automatically align the sentences to subsets of the table cells. Since we are interested in generating natural sentences for single records, we first filter out questions relating to multiple rows if the corresponding SQL query contains aggregate commands (including MAX, MIN, COUNT, SUM, AVG, JOIN, INTERSECT, UNION, GROUP BY, ORDER BY). For the remaining questions relating to a single record in the table, we execute the SQL query to retrieve the answer. We convert the question-answer pair into a declarative sentence, for example:

**Question:** In which year did Greece hold its last Summer Olympics?
**Answer:** 2004

**Declarative Sentence:** Greece held its last Summer Olympics in 2004.

To get the corresponding record, we change the SQL command to SELECT * so we can get the whole row from the table. We then find the corresponding table cells from this row that are from answer columns and WHERE condition columns. The corresponding table cells are then converted into RDF triples in the same way as we described in Section 3.1. In this way, we can get 4,237 sentences with on average two triples for each sentence. Examples of produced declarative sentences can be found in Figure 10.

We employed a similar procedure for WikiTableQuestions, but the questions in WikiTableQuestions often require comparing across different records, so we ended up not including it.

### 3.3 Incorporating Existing Datasets

We incorporate the following existing datasets in the same tripleset-sentence format as discussed before.

**WebNLG 2017** The WebNLG dataset (Gardent et al., 2017a) is a set of triples extracted from DBpedia and the human-annotated target text. We include the WebNLG 2017 dataset consisting of 27731 tripleset-text pairs with up to 7 RDF triples in a triple set covering 15 domains. It contains

---

3We use the rule-based model from https://github.com/kelvinguu/qanli (Demszky et al., 2018). Their neural model code is not released.

4https://gitlab.com/shimorina/webnlg-dataset/-/tree/master/webnlg_challenge_2017
multi-sentential surface realizations and the average number of sentences per realization is 1.44.

**Cleaned E2E** The original E2E dataset (Novikova et al., 2017b) includes dialogue act meaning representations and natural language references in the restaurant domain. The dialog act meaning representations consist of slot-value pairs on eight attributes of restaurants such as name and food. 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. The average number of sentences per realization is 1.59.

To convert a meaning representation (MR) to a tripleset 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 tripleset \{(ALIMENTUM, AREA, CITY CENTRE), (ALIMENTUM, FAMILYFRIENDLY, NO)\}. We drop MRs which do not contain the NAME slot.

### 3.4 Predicate Unification

Combining triples from a wide variety of sources results in multiple predicates that represent the same concept. For example, WebNLG contains the predicate BIRTHDATE while the DART WikiTableQuestions sample contains the predicate DATE OF BIRTH. Additionally, the wide syntactic variety present in Wikipedia tables results in the extraction of different triples that represent the same concept – for example, the DART WikiTableQuestions sample contains both TEAM and CLUB in reference to soccer teams. The triple-to-text task becomes much easier if these disparate predicates are unified, which allows a model to learn a single concept from a single predicate with a greater number of training examples. We manually constructed a predicate mapping table to achieve this. As an example, applying our predicate mapping procedure sends "Hometown," "Home Town," and "Home Town/City" (all of which appear as predicates pre-unification) to the unified predicate "HOMETOWN".

---

\(^5\)https://github.com/tuetschek/e2e-cleaning

### 3.5 Entity Mapping and Delexicalization

Castro Ferreira et al. (2018) pointed out the benefits of delexicalization for NLG models as it allows the model to describe categories instead of noisy entity names (e.g. "President" instead of "Barack Obama").

We perform automatic delexicalization through the named entity extraction interface provided by spaCy.\(^6\) First, we check if spaCy’s "xx_ent_wiki_sm" named entity recognition (NER) model returns a match for the extracted entity. If no match is found, we fall back to spaCy’s “en_core_web_lg” NER model. If this fails too, we check the WebNLG delexicalization for a match. Finally, if there are no matches found, we map the entity to an unknown token.

### 3.6 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 tripleset-sentence examples. This can create models that simply memorize the training data. 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 unseen tables on test split. We first sample some tables as a seed test set, and then compute Jaccard similarity with other tables on table titles and table headers. If other tables have a Jaccard similarity greater than 0.5 with any of the tables in the seed 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 62659/6980/12552 sentences in the train/dev/test sets, respectively.

### 4 Experimental Results

We conducted experiments on DART v1.0.0b and the WebNLG 2017 challenge dataset.

#### 4.1 Models

We investigate several state-of-the-art data-to-text generation models.

**Seq-to-Seq with Attention** We use an encoder-decoder architecture with attention mechanism (Sutskever et al., 2014; Bahdanau et al., 2015).

---

\(^6\)https://github.com/explosion/spaCy
Table 4: Model results on the test set of DART v1.0.0b. ↑: Higher is better. ↓: Lower is better. BART performs the best due to its generalization ability from pretraining.

| Model                     | BLEU ↑ | METEOR ↑ | TER ↓ | MoverScore ↑ | BERTScore(F1) ↑ | BLEURT ↑ |
|---------------------------|--------|----------|-------|--------------|-----------------|----------|
| End-to-End Transformer    | 19.87  | 0.26     | 0.65  | 0.28         | 0.87            | -0.20    |
| Seq-to-Seq with Attention | 29.60  | 0.28     | 0.62  | 0.32         | 0.90            | -0.11    |
| BART                      | 37.06  | 0.36     | 0.57  | 0.44         | 0.92            | 0.22     |

Table 5: The WebNLG 2017 results on the test set. †: We report results from Zhao et al. (2020a) who use the evaluation scripts that are strictly the same as the official challenge resulting in different numbers from the original papers. ‡: Results of our replications; we include previously unreported seen vs. unseen results from Moryossef et al. (2019b) by running evaluation scripts on their original output.

The architecture is a 2-layer bidirectional-LSTMs for the encoder and 300-d word embeddings, 512 hidden units, and 0.3 dropout for the decoder. The input is sequential, formed by concatenating the triples. Delexicalization is applied on the input triple sequence as well as the reference output sentences in accordance with the WebNLG dataset delexicalization. This baseline did not use pretrained word vectors.

**Transformer** We use the Transformer model (Vaswani et al., 2017), previously used by Castro Ferreira et al. (2019) on the WebNLG dataset. We use the end-to-end transformer model without explicit intermediate representations to directly generate text from an unordered (linearized) RDF triple set. The pipeline transformer requires more annotation efforts for additional gold-standard representations of traditional pipeline steps, so we did not use it.

**BART** BART is a pretrained sequence-to-sequence language model using a standard Transformer-based architecture and has shown very effective results for text generation tasks such as machine translation, summarization, conversation response generation, and abstractive QA in Lewis et al. (2020). We fine-tuned the BART-large model with a max output length of 128 and a learning rate of 3e-05 using the same linearization and delexicalization on RDF triple sets as described above.

Furthermore, we also tried BestPlan (Moryossef et al., 2019b) and NeuralPlan (Moryossef et al., 2019a) models, both of which use a step-by-step approach with a text-planning stage followed by a plan-realization stage on the WebNLG dataset. However, the training and inference were quite slow on DART so we could not get these results.

4.2 Evaluation Metrics

We use a variety of automatic metrics to evaluate the quality of the generated text. We report BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), and TER (Snover et al., 2005) which are also used in the official WebNLG challenge. Furthermore, we also use MoverScore (Zhao et al., 2019), BERTScore (Zhang et al., 2020a), and
BLEURT (Sellam et al., 2020), which are new metrics that incorporate semantics rather than surface forms using contextual embeddings. We did not use PARENT (Dhingra et al., 2019) because not all of our inputs are in a tabular form (e.g., the ones from WebNLG and E2E).

4.3 Results

**DART** Our experimental results on DART are summarized in Table 4. The BART model has the highest performance among three models with a BLEU score of 37.06. This is attributed to BART’s generalization ability due to pretraining. However, language models such as BART 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). In addition, while the pretrained language 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. Furthermore, the end-to-end transformer has the lowest performance since the transformer model needs intermediate pipeline planning steps to have higher performance. Similar findings can be found in Castro Ferreira et al. (2019).

**WebNLG** Furthermore, we also investigate if DART can be used to improve the performance on other data-to-text generation tasks. To this end, we finetune the BART model on the WebNLG 2017 challenge, and gradually augment the training data with the train split of DART. The experimental results can be found in Table 5. Our finetuned BART model achieves 39.05 BLEU scores on all categories, while adding DART as additional training data can improve it to 45.89. The improvement mainly comes from the unseen categories increasing from 26.23 to 37.85, demonstrating that DART is particularly helpful for out-of-domain generalization due to its open-domain nature.

5 Conclusion

In this paper, we introduce DART as a large and open-domain corpus for structured data record to text generation. DART’s hierarchical and structured format differentiates itself from other open-domain table-to-text corpora such as ToTTo. 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. For future work, we will explore more controlled and high-fidelity generation using pretrained language models conditioned on semantic representations with encoding methods incorporating the ontology hierarchy.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In ICLR.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for mt evaluation with improved correlation with human judgments. In ACL Workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization.

Eva Banik, Claire Gardent, and Eric Kow. 2013. The KBGen challenge. In Proceedings of the 14th European Workshop on Natural Language Generation.

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.

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

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.

Thiago Castro Ferreira, Diego Moussallem, Emiel Krahmer, and Sander Wubben. 2018. Enriching the WebNLG corpus. In INLG.

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, Yuncai Zhang, Sairam Sundaresan, and William Yang Wang. 2020b. Logic2text: High-fidelity natural language generation from logical forms. arXiv preprint arXiv:2004.14579.

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

Julian Chow, Lucia Specia, and Pranava Madhyastha. 2019. WMDO: Fluency-based word mover’s distance for machine translation evaluation. In Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1).

Marco Damonte and Shay B Cohen. 2019. Structural neural encoders for amr-to-text generation. In NAACL.

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

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In Proceedings of the EACL 2014 Workshop on Statistical Machine Translation.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL.

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.

Bayu Distiawan, Jianzhong Qi, Rui Zhang, and Wei Wang. 2018. Gtr-lstm: A triple encoder for sentence generation from rdf data. In ACL.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In NeurIPS.

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

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG Challenge. 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. arXiv preprint arXiv:1910.08435.

Thiago Castro Ferreira, Iacer Calixto, Sander Wubben, and Emiel Krahmer. 2017. Linguistic realisation as machine translation: Comparing different mt models for amr-to-text generation. In INLG.

Jeffrey Flanigan, Chris Dyer, Noah A Smith, and Jaime G Carbonell. 2016. Generation from abstract meaning representation using tree transducers. In NAACL.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017a. Creating training corpora for NLG microplanners. In ACL.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017b. 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.

Eli Goldberg, Norbert Driedger, and Richard I Kittredge. 1994. Using natural-language processing to produce weather forecasts. IEEE Expert, 9(2):45–53.

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*.

Hayate Iso, Yui Uehara, Tatsuya Ishigaki, Hiroshi Noji, Eiji Aramaki, Ichiro Kobayashi, Yusuke Miyao, Naoaki Okazaki, and Hiroya Takamura. 2019. Learning to select, track, and generate for data-to-text. In *ACL*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *ACL*.

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

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *NAACL*.

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

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *ICML*.

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, Ming Shen, Wangchunshu Zhou, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2019. Commongen: A constrained text generation challenge for generative commonsense reasoning. *CoRR*, abs/1911.03705.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

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

Chi-kiu Lo. 2019. Yisi-a unified semantic mt quality evaluation and estimation metric for languages with different levels of available resources. In *Proceedings of the Fourth Conference on Machine Translation*.

Diego Marcheggiani and Laura Perez-Beltrachini. 2018. Deep graph convolutional encoders for structured data to text generation. In *INLG*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *NAACL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019a. Improving quality and efficiency in plan-based neural data-to-text generation. In *INLG*.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019b. 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 *SIGHJOUR*.

Michael Oâ€™Donnell, Alistair Knott, Jon Oberlander, and Chris Mellish. 2000. Optimising text quality in generation from relational databases. In *INLG*.
Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In ACL.

Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. arXiv preprint arXiv:2004.14373.

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

Laura Perez-Beltrachini and Mirella Lapata. 2018. Bootstrapping generators from noisy data. In NAACL.

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. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv preprint arXiv:1910.10683.

Nazneen Fatema Rajani, Rui Zhang, Yi Chern Tan, Stephan Zheng, Jeremy Weiss, Amrit Vyas, Abhijit Gupta, Caiming Xiong, Richard Socher, and Dragomir Radev. 2020. ESPRIT: Explaining Solutions to Physical Reasoning Tasks. In ACL.

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

Ehud Reiter, Somayajulu Sripada, Jim Hunter, Jin Yu, and Ian Davy. 2005. Choosing words in computer-generated weather forecasts. Artificial Intelligence, 167(1-2):137–169.

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

Leonardo FR Ribeiro, Yue Zhang, Claire Gardent, and Iryna Gurevych. 2020. Modeling global and local node contexts for text generation from knowledge graphs. Transactions of the Association for Computational Linguistics.

Marco Roberti, Giovanni Bonetta, Rossella Cancelliere, and Patrick Gallinari. 2019. Copy mechanism and tailored training for character-based data-to-text generation. In ECML-PKDD.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. Transactions of the Association for Computational Linguistics.

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

Lei Sha, Lili Mou, Tianyu Liu, Pascal Poupart, Sujuan Li, Baobao Chang, and Zhifang Su. 2018. Order-planning neural text generation from structured data. In AAAI.

Hamidreza Shahidi, Ming Li, and Jimmy Lin. 2020. Two birds, one stone: A simple, unified model for text generation from structured and unstructured data. In ACL.

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 2019. Pragmatically informative text generation. In NAACL.

Xiaoyu Shen, Ernie Chang, Hui Su, Jie Zhou, and Dietrich Klakow. 2020. Neural data-to-text generation via jointly learning the segmentation and correspondence. In ACL.
Anastasia Shimorina and Claire Gardent. 2018. Handling rare items in data-to-text generation. In INLG.

Mathew Snover, Bonnie Dorr, Richard Schwartz, John Makhoul, Linnea Micciulla, and Ralph Weischedel. 2005. A study of translation error rate with targeted human annotation. In AMTA.

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

Lin Feng Song, Ante Wang, Jinsong Su, Yue Zhang, Kun Xu, Yubin Ge, and Dong Yu. 2020. Structural information preserving for graph-to-text generation. In ACL.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In NeurIPS.

Kumiko Tanaka-Ishii, Kôiti Hasida, and Itsuki Noda. 1998. Reactive content selection in the generation of real-time soccer commentary. In COLING.

Yasufumi Taniguchi, Yukun Feng, Hiroya Takamura, and Manabu Okumura. 2019. Generating live soccer-match commentary from play data. In AAAI.

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.

Hongmin Wang. 2019. Revisiting challenges in data-to-text generation with fact grounding. In INLG.

Qingyun Wang, Xiaoman Pan, Lifu Huang, Boliang Zhang, Zhiying Jiang, Heng Ji, and Kevin Knight. 2018. Describing a knowledge base. In INLG.

Tianming Wang, Xiaojun Wan, and Hanqi Jin. 2020a. Amr-to-text generation with graph transformer. Transactions of the Association for Computational Linguistics, 8:19–33.

Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. 2020b. Towards faithful neural table-to-text generation with content-matching constraints. In ACL.

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.

Rong Ye, Wenxian Shi, Hao Zhou, Zhongyu Wei, and Lei Li. 2020. Variational template machine for data-to-text generation. In ICLR.

Tao Yu, Rui Zhang, Heyang Er, Su Yi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In EMNLP.

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

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020b. DialoGPT: Large-scale generative pre-training for conversational response generation. In ACL.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020a. 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.

Yanbin Zhao, Lu Chen, Zhi Chen, Ruisheng Cao, Su Zhu, and Kai Yu. 2020b. Line graph enhanced amr-to-text generation with mix-order graph attention networks. In ACL.

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

Jie Zhu, Junhui Li, Muhua Zhu, Longhua Qian, Min Zhang, and Guodong Zhou. 2019a. Modeling graph structure in transformer for better AMR-to-text generation. In EMNLP.

Yaoming Zhu, Juncheng Wan, Zhiming Zhou, Liheng Chen, Lin Qiu, Weinan Zhang, Xin Jiang, and Yong Yu. 2019b. Triple-to-text: Converting rdf triples into high-quality natural languages via optimizing an inverse kl divergence. In SIGIR.
Figure 4: A WikiTableQuestions table that uses [TITLE] in the ontology.

| Year   | [TITLE]: 5150 Album |
|--------|--------------------|
| Single | Year               |
| 1986   | Best of Both Worlds |
| 1986   | Dreams             |
| 1986   | Love Walks In      |

Figure 5: A manually annotated table from WikiTableQuestions with a sentence that uses the table title.

Figure 6: An example of the data cleaning. The top left table had a missing column and the table title was not specific to the data; the bottom left table had repeat columns in the table. Internal annotators went through malformed tables and corrected them.

Table Title: Peter De Villiers

| Competition | Event | Competition Venue | [TITLE] | Position | Event | [TITLE] | Time |
|-------------|-------|-------------------|---------|----------|-------|---------|------|
| 2000        | World Junior Championships | Santiago, Chile | 2nd | 400 m hurdles | 50.52 |
| 2000        | World Junior Championships | Santiago, Chile | 8th | 4x400 m relay | 3:07.66 |
| 2008        | World Championships | Helsinki, Finland | 17th | 400 m hurdles | 49.75 |
| 2005        | World Championships | Helsinki, Finland | 13th | 4x400 m relay | 3:04.64 |
| 2006        | Commonwealth Games | Melbourne, Australia | 7th | 400 m hurdles | 50.51 |
| 2007        | African Championships | Durban, Mauritius | 6th | 400 m hurdles | 49.96 |
| 2007        | All-Africa Games | Algiers, Algeria | 8th | 4x400 m relay | DNF |
| 2007        | World Championships | Osaka, Japan | 15th | 400 m hurdles | 49.37 |
| 2008        | Olympic Games | Beijing, China | 12th | 400 m hurdles | 49.44 |

The competitor won a silver medal at the 2000 World Junior Championships in Santiago, Chile.

They finished the 400 m hurdles 7th in 50.51.

The 2007 All-Africa Games held a 4x400 m relay.

He competed in the 2008 Beijing Olympic Games.
Chivas and América will compete in the semifinals of the Apertura 2006 tournament.

Terry Jenkins lost the game with Per Laursen in the 1st Round of 2014 PDC World Darts Championship.

Figure 8: Examples of triple sets and sentences.

| Title: 1930 Indianapolis 500 |
|-----------------------------|
| Pos | No | Driver | Constructor | Lap | Cup |
| 7 | 69 | Duke Dimmow | Kurtis Kraft-Offenhauser | 4:34.67 | 6.7 |

**Title:** Kurt Muschler Award

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

**Title:** Maxi Priest

| Year | Song | U.S. | U.S. R&B | U.K | Album |
|------|------|------|---------|-----|-------|
| 1990 | "Just a Little Bit Longer" | 62 | 30 | None | Bonafide |

**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.

**Title:** Cincinnati Kings

| Year | Division | League | Regular Season | Playoffs | Open Cup |
|------|----------|--------|----------------|----------|----------|
| 2008 | 4 | usl pdl | 4th | did not qualify | did not qualify |

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

**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.