# Ambiguity Detection and Textual Claims Generation from Relational Data

(Discussion Paper)

Enzo Veltri[1], Donatello Santoro[1], Gilbert Badaro[2], Mohammed Saeed[2] and Paolo Papotti[2]

[1] *Università degli Studi della Basilicata (UNIBAS), Potenza, Italy*
[2] *EURECOM, Biot, France*

### Abstract

Computational fact checking, (given a textual claim and a table, verify if the claim holds w.r.t. the given data) and data-to-text generation (given a subset of cells, produce a sentence describing them) exploit the relationship between relational data and natural language text. Despite promising results in these areas, state-of-the-art solutions simply fail in managing "data-ambiguity", i.e., the case when there are multiple interpretations of the relationship between the textual sentence and the relational data. To tackle this problem, we present Pythia, a system that, given a relational table $D$, generates textual sentences that contain factual ambiguities w.r.t. the data in $D$. Such sentences can then be used to train target applications in handling data-ambiguity.

In this paper, we discuss how Pythia generates data ambiguous sentences for a given table in an unsupervised fashion using data profiling and query generation. We then show how two existing downstream applications, namely data-to-text and computational fact checking, benefit from Pythia's generated sentences, improving the state-of-the-art results without manual user effort.

### Keywords

Unsupervised Text Generation, Fact Checking, Data to Text Generation, Data Ambiguity

## 1. Introduction

Ambiguity is common in natural language in many forms [1, 2, 3]. We focus on the ambiguity of a factual sentence w.r.t. the data in a table, or relation. The problem of data ambiguities in sentences is relevant for many natural language processing (NLP) applications that use relational data, from computational fact checking [4, 5, 6] to data-to-text generation [7, 8], and question answering in general [9, 10].

Consider a fact checking application, where the goal is to verify a textual claim against relational data, and the sentence "Carter LA has higher shooting than Smith SF" to be checked against the dataset $D$ in Table 1. Even as humans, it is hard to state if the sentence is true or false, w.r.t. the data in $D$. The challenge is due to the two different meanings that can be

*SEBD 2022: The 30th Italian Symposium on Advanced Database Systems, June 19-22, 2022, Tirrenia (PI), Italy*

enzo.veltri@unibas.it (E. Veltri); donatello.santoro@unibas.it (D. Santoro); gilbert.badaro@eurecom.fr (G. Badaro); mohammed.saeed@eurecom.fr (M. Saeed); papotti@eurecom.fr (P. Papotti)
0000-0001-9947-8909 (E. Veltri); 0000-0002-5651-8584 (D. Santoro); 0000-0002-7277-617X (G. Badaro); 0000-0001-6815-9374 (M. Saeed); 0000-0003-0651-4128 (P. Papotti)

© 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

|  | Player | Team | FG% | 3FG% | fouls | apps |
|---|---|---|---|---|---|---|
| $t_1$ | Carter | LA | 56 | 47 | 4 | 5 |
| $t_2$ | Smith | SF | 55 | 50 | 4 | 7 |
| $t_3$ | Carter | SF | 60 | 51 | 3 | 3 |

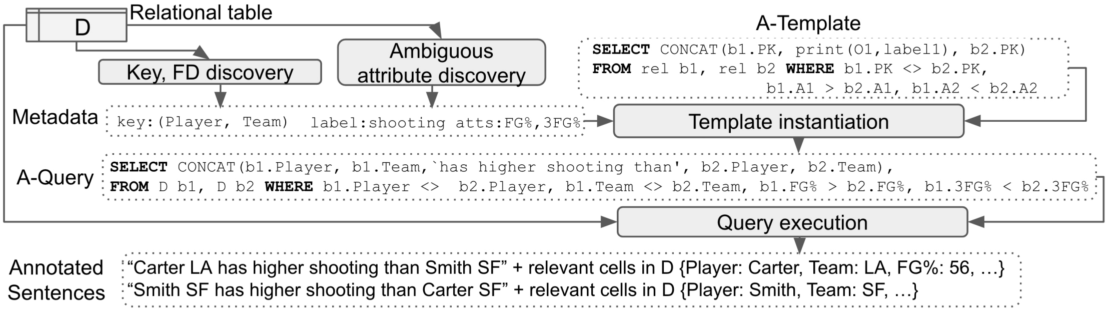
**Table 1**
Dataset $D$. The sentence “Carter has higher shooting than Smith” is data ambiguous w.r.t. $D$.

matched to *shooting percentage*: the claim can refer to statistics about *Field Goal* (**FG%**) or *3-point Field Goal* (**3FG%**). We refer to this issue as *data ambiguity*, i.e., the existence of more than one unique interpretation of a factual sentence w.r.t. the data for a human reader.

Sentences with data ambiguities are not present in existing corpora with sentences annotated w.r.t. the data, which leads target applications to simply fail in these scenarios. In existing fact checking applications, the sentence above leads to a single interpretation, that is incorrect in 50% of the cases. This problem is important in practice. In the log of an online application for fact checking [1], we found that more than 20 thousand claims submitted by the users are data ambiguous, over 80% of the submitted sentences!

While traditionally high quality corpora of annotated text sentences come from extensive and expensive manual work, in this work we demonstrate that by exploiting deep learning over the input table, we can automatically generate SQL scripts that output data ambiguous annotated sentences. SQL enables the efficient generation of a large number of annotated sentences for a given table, which in turn can be used as training data to significantly advance the target downstream applications.

In this short paper we describe Pythia, a system for the automatic generation of sentences with data ambiguities, which effectively solves the problem of crafting training examples at scale [11, 12]. More precisely, we first introduce our solution based on data profiling, query generation, and a novel algorithm to detect ambiguous attributes (Section 2). We then illustrate how the users engage with Pythia with interactive parameters, including the ability to load their own datasets and automatically obtain data ambiguous sentences. We finally show how the generated sentences are used as training data to improve the coverage and quality of two target applications (Section 3).



```
SELECT CONCAT(b1.PK, print(O1,label1), b2.PK)
FROM rel b1, rel b2 WHERE b1.PK <> b2.PK,
                          b1.A1 > b2.A1, b1.A2 < b2.A2
```

```
key:(Player, Team)  label:shooting atts:FG%,3FG%
```

```
SELECT CONCAT(b1.Player, b1.Team,`has higher shooting than', b2.Player, b2.Team),
FROM D b1, D b2 WHERE b1.Player <>  b2.Player, b1.Team <> b2.Team, b1.FG% > b2.FG%, b1.3FG% < b2.3FG%
```

“Carter LA has higher shooting than Smith SF” + relevant cells in D {Player: Carter, Team: LA, FG%: 56, …}
“Smith SF has higher shooting than Carter SF” + relevant cells in D {Player: Smith, Team: SF, …}

**Figure 1:** Pythia takes as input any relational table and generates data ambiguous sentences.

[1] https://coronacheck.eurecom.fr

## 2. System Overview

Consider again the sentence about the higher shooting percentage. The sentence can be obtained from a SQL query over $D$ comparing the **FG%** and **3FG%** attributes for pairs of players.

```
q1: SELECT CONCAT(b1.Player, b1.Team, 'has higher
          shooting than', b2.Player, b2.Team)
    FROM D b1, D b2
    WHERE b1.Player <> b2.Player AND
          b1.Team <> b2.Team AND
          b1.FG% > b2.FG% AND b1.3FG% < b2.3FG%
```

Another example is "Carter has higher 3FG% than Smith". There are two players named 'Carter' and the sentence is true for the one in team 'SF', but it is false for the one in team 'LA'. This sentence can also be generated with a query over $D$.

Given a relation, the key idea is to model a sentence as the result of a query over the data. However, coming up with the query automatically from a given relational table is challenging. Consider again query $q1$ above. First, the ambiguous attribute pair is not given with the table, e.g., **FG%** and **3FG%** in $q1$ must be identified. Second, an ambiguous sentence requires a *label*, words in the Select clause that plausibly refer to the two attributes, e.g., "shooting" in $q1$ is the label for **FG%** and **3FG%**. To handle these challenges, we restrict the query generation search space by making use of templates, which get instantiated with deep learning models predicting the query parameters from the given table, as detailed next.

**Data Ambiguity with A-Queries.** Sentences can be data ambiguous in different ways. For example, the claim $s_1$: "Carter LA has better shooting than Smith SF" is ambiguous w.r.t. relation $D$ in Table 1 because even as humans we cannot state for sure if it refers to regular Field Point statistics or performance for the 3 point range. This is an example of *attribute ambiguity* over two attributes, but the same sentence can be ambiguous to more than two attributes.

Sentence $s_2$: "Carter committed 4 fouls" shows a different kind of data ambiguity. As rows are identified by two attributes in $D$, a reference to only part of the composite key makes it impossible to identify the right player. This is a *row ambiguity* over two rows, but the same sentence can be ambiguous to more rows.

A sentence can also be ambiguous in the two dimensions, both over the attributes and the rows. With more dimensions, also the number of possible interpretations increases. We therefore distinguish between *attribute*, *row*, and *full* for the structure of the ambiguity. We further distinguish between *contradictory* and *uniform* sentences. The first type leads to interpretations that have opposite factual match w.r.t. the data. Sentences $s_1$ and $s_2$ are both contradictory, as one interpretation is confirmed from the data and the other is refuted. For uniform sentences, all interpretations have equal matching w.r.t. the data, e.g., sentence "Carter has more appearances than Smith" is refuted for both Carter players in $D$. We found the distinction between *contradictory* and *uniform* sentences crucial in target applications, as they guarantee more control over the output corpora.

An *ambiguity query* (or *a-query*) executed over table $D$ returns $s_1, \ldots, s_n$ sentences with data ambiguity w.r.t. $D$. Consider again $q1$ defined over $D$, it returns all the pairs of distinct players that lead to contradictory sentences. The same query can be modified to obtain uniform sentences by changing one operator in one of the WHERE clause comparing statistics, e.g., change '$<$' to '$>$' in the last clause. Finally, by removing the last two WHERE clauses, the query returns both contradictory and uniform sentences.

**From the Table to the Sentences.** A-queries can be obtained from parametrized SQL *A-Query Templates.* The parameters identify the elements of the query and can be assigned to (i) relation and attribute names, (ii) a set of labels, i.e. words with a meaning that applies for two or more attributes, (iii) comparison operators.

Given an A-Query Template is possible to generate multiple a-queries. Pythia already stores multiple handcrafted SQL templates that cover different ambiguity types and can be widely used with different datasets, but it also allows users to add other custom templates using a predefined syntax.

Figure 2 summarizes the overall process. Starting from dataset $D$, key/FDs profiling methods and our ambiguous attribute discovery algorithm identify the *metadata* to fill up the *A-Templates* provided in Pythia. All compatible operators are used by default. Metadata, operators, and templates can be refined and extended manually by users. Once the template is instantiated, the *A-Query* is ready for execution over $D$ to obtain the data ambiguous sentences, each annotated with the relevant cells in $D$.

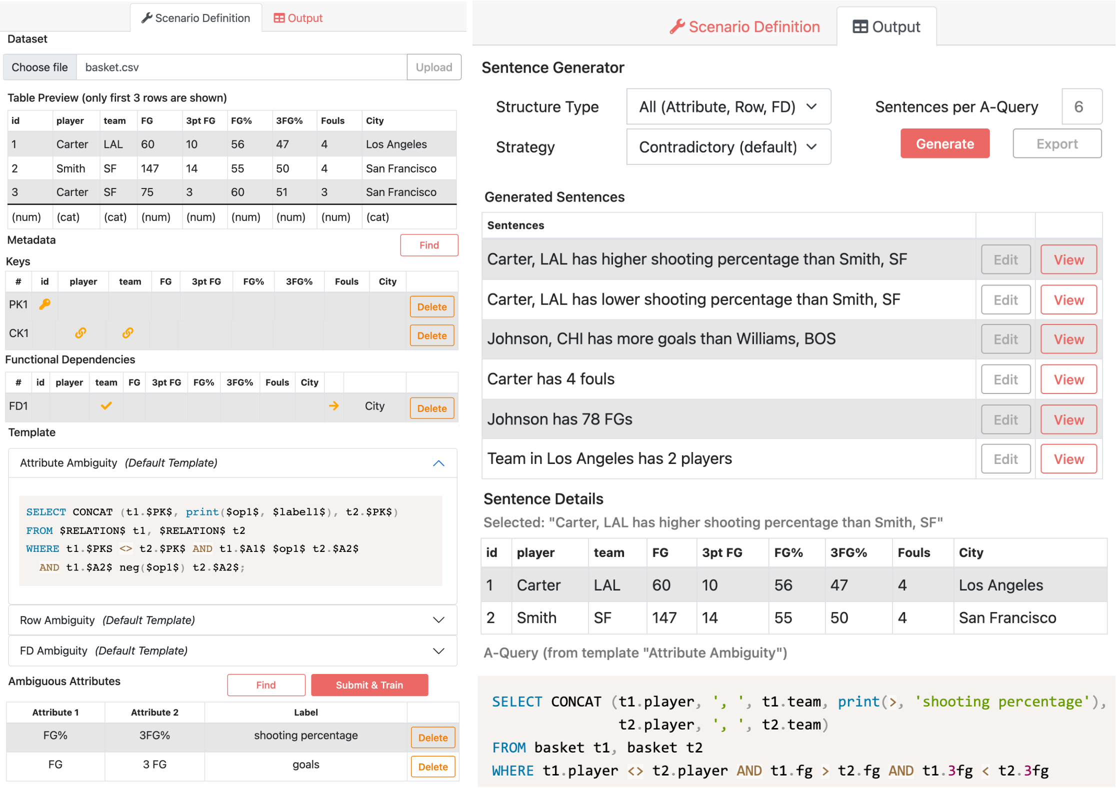
Parameters values to instantiate the templates are automatically populated by Pythia using existing profiling methods and a novel deep learning method to find attribute ambiguities with the corresponding ambiguous words. We remark our focus on generating sentences with genuine data ambiguity, which is in contrast to other cases where the correct meaning of the text is entirely clear to a human but an algorithm detects more than one interpretation in the data. For example, "Carter SF has played 3 times" clearly refers to the player *Appearances* (**apps**), but an algorithm could be uncertain between attributes **fouls** and **apps**. We refer the reader to the full paper for details about our deep learning solution [12].

## 3. Experiments

The section is organized in two parts. In the first part, we focus on the data ambiguity sentences from a given table. In the second part, we show how the generated corpus can improve three target applications.

### 3.1. Dataset Generation

Pythia provides a visual interface that allows to use preloaded text generation scenarios or create a new one. Figure 2a shows the configuration of a new scenario. First, the users uploads a dataset. Then metadata such as primary keys, composite keys and FDs can be manually provided or obtained with profiling [13]. The core of Pythia is built on top of A-Query Templates. Pythia comes with a set of three generic default templates that were manually crafted from the analysis of an online fact checking application's log. The three templates cover more than 90% of user submitted claims with data ambiguities. More A-Query Templates can be provided by the user. Parameters' values of the templates are automatically populated by Pythia using the schema metadata and a novel deep learning module to find ambiguous attributes and the word that



Dataset

basket.csv

Table Preview (only first 3 rows are shown)

| id | player | team | FG | 3pt FG | FG% | 3FG% | Fouls | City |
|---|---|---|---|---|---|---|---|---|
| 1 | Carter | LAL | 60 | 10 | 56 | 47 | 4 | Los Angeles |
| 2 | Smith | SF | 147 | 14 | 55 | 50 | 4 | San Francisco |
| 3 | Carter | SF | 75 | 3 | 60 | 51 | 3 | San Francisco |
| (num) | (cat) | (cat) | (num) | (num) | (num) | (num) | (num) | (cat) |

Metadata

Keys

| # | id | player | team | FG | 3pt FG | FG% | 3FG% | Fouls | City | |
|---|---|---|---|---|---|---|---|---|---|---|
| PK1 | 🔑 | | | | | | | | | Delete |
| CK1 | | 🔗 | 🔗 | | | | | | | Delete |

Functional Dependencies

| # | id | player | team | FG | 3pt FG | FG% | 3FG% | Fouls | City | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FD1 | | | ✓ | | | | | | | → | City | Delete |

Template

Attribute Ambiguity *(Default Template)*

```
SELECT CONCAT (t1.$PK$, print($op1$, $label1$), t2.$PK$)
FROM $RELATION$ t1, $RELATION$ t2
WHERE t1.$PKS <> t2.$PK$ AND t1.$A1$ $op1$ t2.$A2$
  AND t1.$A2$ neg($op1$) t2.$A2$;
```

Row Ambiguity *(Default Template)*

FD Ambiguity *(Default Template)*

Ambiguous Attributes

| Attribute 1 | Attribute 2 | Label | |
|---|---|---|---|
| FG% | 3FG% | shooting percentage | Delete |
| FG | 3 FG | goals | Delete |

(a) Pythia configuration

Sentence Generator

Structure Type: All (Attribute, Row, FD)

Sentences per A-Query: 6

Strategy: Contradictory (default)

Generated Sentences

| Sentences | | |
|---|---|---|
| Carter, LAL has higher shooting percentage than Smith, SF | Edit | View |
| Carter, LAL has lower shooting percentage than Smith, SF | Edit | View |
| Johnson, CHI has more goals than Williams, BOS | Edit | View |
| Carter has 4 fouls | Edit | View |
| Johnson has 78 FGs | Edit | View |
| Team in Los Angeles has 2 players | Edit | View |

Sentence Details

Selected: "Carter, LAL has higher shooting percentage than Smith, SF"

| id | player | team | FG | 3pt FG | FG% | 3FG% | Fouls | City |
|---|---|---|---|---|---|---|---|---|
| 1 | Carter | LAL | 60 | 10 | 56 | 47 | 4 | Los Angeles |
| 2 | Smith | SF | 147 | 14 | 55 | 50 | 4 | San Francisco |

A-Query (from template "Attribute Ambiguity")

```
SELECT CONCAT (t1.player, ', ', t1.team, print(>, 'shooting percentage'),
               t2.player, ', ', t2.team)
FROM basket t1, basket t2
WHERE t1.player <> t2.player AND t1.fg > t2.fg AND t1.3fg < t2.3fg
```

(b) Output visualization

**Figure 2:** Pythia configuration of a scenario and output visualization with data ambiguous sentences

describes both attributes (label). Pythia allows the user to submit more ambiguous attributes and labels, ultimately submitting feedback that further improves the underlying ML module.

After the scenario's definition, the user generates ambiguous sentences as depicted in Figure 2b. To control the type of ambiguities of the generated dataset, it is possible to select the structure type, i.e., the A-Query Templates to use, strategies (Contradictory, Uniform True, Uniform False), and the number of output sentences per a-query. The system outputs for every generated sentence the pair of template and a-query that generated it, together with the data in the table that relates to the sentence. In this step, the users can also fine-tune templates and queries interactively. The generated corpus can then be exported in CSV, JSON, and application specific formats.

## 3.2. Use cases

In this section we use seven datasets. We use Iris, Abalone, Adults and Mushroom from UCI repository [14], two versions of Basket dataset, one with acronyms and one with full names for the attributes [15], and a version of Soccer crawled from the web. Training data for downstream applications is generated using three datasets (Iris, Basket full names and Soccer), the remaining datasets are used for test data. For each dataset, we generate sentences with contradictory, uniform true and uniform false strategies. We denote with $P_t$ and $P_d$ the training and developing (testing) data generated with Pythia, respectively.

| | BLEU | ROUGE |
|---|---|---|
| Original (**ToTTo** [7]) | 0.547 | 0.202 |
| **ToTTo** fine-tuned with Pythia's output | **37.631** | **0.631** |

**Table 2**
Impact of $P_t$ data on table-to-text generation.

**ToTTo**. ToTTo [7] is a dataset for table-to-text generation, i.e., given a set of cells, the goal is to generate a sentence describing the input data. ToTTo contains annotated sentences that involve reasoning and comparisons among rows, columns, or cells. We test a T5 model[16] trained on the original ToTTo for sentence generation. The results show that ambiguities are not handled in ToTTo, and thus, the model simply fails when it makes predictions over $P_d$. We use the $P_t$ to fine-tune the model. Such fine-tuning step improves the performance of the model in terms of BLEU and ROUGE metrics according to the results in Table 2 (higher is better).

| | **CLASS** | **P** | **R** | **F1** |
|---|---|---|---|---|
| **Feverous Baseline** | Ambiguous | n.a. | n.a. | n.a. |
| | Supports | 0.33 | 1.00 | 0.49 |
| | Refutes | 0.00 | 0.00 | 0.00 |
| **Feverous Baseline trained on $P_t$** | Ambiguous | 0.92 | 0.85 | 0.89 |
| | Supports | 0.96 | 0.95 | 0.95 |
| | Refutes | 0.88 | 0.95 | 0.92 |

**Table 3**
Impact of $P_t$ data on fact checking (Feverous).

**Feverous**. Feverous [17] is a dataset for fact checking containing textual claims. Every claim is annotated with the data identified by human annotators to *Support* or *Refute* a claim. Unfortunately, a model trained with the Feverous baseline pipeline [2] fails on the classification of an ambiguous claim from $P_d$, as the *Ambiguous* label is not present in the original dataset. Training the same model with $P_t$, which contains also examples of ambiguity, radically improves the performance for all classes. Table 3 reports the results in terms of precision, recall, and f-measure.

**CoronaCheck**. As a second fact checking application, we show the verification of statistical claims related to COVID-19 [5]. We incorporate the contributions of Pythia to improve an existing system [3]. The original system was not able to handle attribute ambiguity. Also, row

[2] https://github.com/Raldir/FEVEROUS
[3] https://coronacheck.eurecom.fr

ambiguity caused the original system to hallucinate with lower classification accuracy for ambiguous claims. One of the tables used for verifying the statistical claims consists of the following attributes, among others: *country*, *date*, *total_confirmed*, *new_confirmed*, *total_fatality_rate*, *total_mortality_rate*. From the analysis of the log of claims submitted by users, an example of common attribute ambiguity is between attributes *total_fatality_rate* and *total_mortality_rate* for sentences containing "death rate". Another example of attribute ambiguity is between *total_confirmed* and *new_confirmed* when sentences that only mention "cases" are verified. Examples of row ambiguity consist of claims that refer to two records with the same location but different timestamps, such as "In France, 10k confirmed cases have been reported" (today or yesterday?).

| | **Users' Claims** | **Accuracy original** | **Accuracy original+** PYTHIA |
|---|---|---|---|
| **Row Amb.** | 44 | 25/44 | **36**/44 |
| **Attribute Amb.** | 9 | 0/9 | **6**/9 |
| **No Ambiguity** | 6 | 6/6 | **6**/6 |
| **Total** | 50 | 31/50 | **42**/50 |

**Table 4**
Impact of $P_t$ data on fact checking (CoronaCheck).

PYTHIA enabled us to improve CoronaCheck by automatically generating ambiguity aware training data. The new corpora have been used to train new classifiers that allowed the extension of the original system. We consider 50 claims from the log of CoronaCheck and we study the types of ambiguities they include. As shown in Table 4, 88% include a row or attribute ambiguities. 83% of those ambiguous claims have row ambiguity, and 17% have both attribute and row ambiguity. The original CoronaCheck fails to classify claims with ambiguity. However, training the system with the output of PYTHIA leads to handling most of the row and attribute ambiguity. To show the benefit of using PYTHIA compared to the original system, we consider the following example: "June 2021: France has 111,244 Covid-19 deaths". *(True for total_deaths, False for new_deaths).* The original system returns a false decision by checking against *new_deaths* only. With PYTHIA support, the decision is True for *total_deaths* and False otherwise.

## References

[1] [n.d.], Notes on ambiguity, https://cs.nyu.edu/~davise/ai/ambiguity.html, 2021.

[2] D. Newman-Griffis, et al., Ambiguity in medical concept normalization: An analysis of types and coverage in electronic health record datasets, Journal of the American Medical Informatics Association 28 (2021) 516–532.

[3] P. Lapadula, G. Mecca, D. Santoro, L. Solimando, E. Veltri, Humanity is overrated. or not. automatic diagnostic suggestions by greg, ml, in: European Conference on Advances in Databases and Information Systems, Springer, 2018, pp. 305–313.

[4] N. Hassan, et al, Claimbuster: The first-ever end-to-end fact-checking system, Proc. VLDB Endow. 10 (2017) 1945–1948.

[5] G. Karagiannis, M. Saeed, P. Papotti, I. Trummer, Scrutinizer: A mixed-initiative approach to large-scale, data-driven claim verification, Proc. VLDB Endow. 13 (2020) 2508–2521.

[6] Y. Wu, P. K. Agarwal, C. Li, J. Yang, C. Yu, Computational fact checking through query perturbations, ACM Trans. Database Syst. 42 (2017) 4:1–4:41.

[7] A. P. Parikh, X. Wang, S. Gehrmann, M. Faruqui, B. Dhingra, D. Yang, D. Das, Totto: A controlled table-to-text generation dataset, in: EMNLP, ACL, 2020, pp. 1173–1186.

[8] I. Trummer, Data vocalization with CiceroDB, in: CIDR, www.cidrdb.org, 2019.

[9] W. Chen, M. Chang, E. Schlinger, W. Y. Wang, W. W. Cohen, Open question answering over tables and text, in: ICLR, OpenReview.net, 2021.

[10] J. Thorne, M. Yazdani, M. Saeidi, F. Silvestri, S. Riedel, A. Y. Levy, From natural language processing to neural databases, Proc. VLDB Endow. 14 (2021) 1033–1039.

[11] E. Veltri, D. Santoro, G. Badaro, M. Saeed, P. Papotti, Pythia: Unsupervised generation of ambiguous textual claims from relational data, in: SIGMOD, 2022. doi:10.1145/3514221.3520164.

[12] E. Veltri, G. Badaro, M. Saeed, P. Papotti, Pythia: Generating Ambiguous Sentences From Relational Data, https://www.eurecom.fr/~papotti/pythiaTr.pdf, 2021.

[13] Z. Abedjan, L. Golab, F. Naumann, T. Papenbrock, Data Profiling, Synthesis Lectures on Data Management, Morgan & Claypool Publishers, 2018.

[14] D. Dua, C. Graff, UCI machine learning repository, 2017. URL: http://archive.ics.uci.edu/ml.

[15] S. Wiseman, S. Shieber, A. Rush, Challenges in data-to-document generation, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, 2017, pp. 2253–2263. URL: https://aclanthology.org/D17-1239. doi:10.18653/v1/D17-1239.

[16] C. Raffel, et al., Exploring the limits of transfer learning with a unified text-to-text transformer, Journal of Machine Learning Research 21 (2020).

[17] R. Aly, et al., FEVEROUS: Fact extraction and VERification over unstructured and structured information, in: Thirty-fifth Conference on Neural Information Processing Systems (NIPS - Datasets and Benchmarks), 2021.