Pythia: Unsupervised Generation of Ambiguous Textual Claims from Relational Data

Enzo Veltri
enzo.veltri@unibas.it
University of Basilicata

Donatello Santoro
donatello.santoro@unibas.it
University of Basilicata

Gilbert Badaro
gilbert.badaro@eurecom.fr
EURECOM

Mohammed Saeed
mohammed.saeed@eurecom.fr
EURECOM

Paolo Papotti
papotti@eurecom.fr
EURECOM

ABSTRACT

Applications such as computational fact checking and data-to-text generation 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 introduce 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 demonstration, we first show how our system generates data ambiguous sentences for a given table in an unsupervised fashion by data profiling and query generation. We then demonstrate how two existing applications benefit from Pythia's generated sentences, improving the state-of-the-art results. The audience will interact with Pythia by changing input parameters in an interactive fashion, including the upload of their own dataset to see what data ambiguous sentences are generated for it.

ACM Reference Format:
Enzo Veltri, Donatello Santoro, Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2022. Pythia: Unsupervised Generation of Ambiguous Textual Claims from Relational Data. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3514221.3520164

1 INTRODUCTION

Ambiguity is common in natural language in many forms [7]. 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, 6, 11] to data-to-text generation [8, 10], and question answering in general [2, 9].

SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA

ACM ISBN 978-1-4503-9249-5/22/06…$15.00
https://doi.org/10.1145/3514221.3520164

Table 1: Dataset D. The sentence "Carter has higher shooting than Smith" is data ambiguous w.r.t. D.

	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

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 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 (https://coronacheck.eurecom.fr/), 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 demonstration we show Pythia, a system for the automatic generation of sentences with data ambiguities, which effectively solves the problem of crafting training examples at scale. 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 will engage with Pythia with interactive parameters, including the ability to load their own datasets and automatically obtain data ambiguous sentences. We will also demonstrate how the generated sentences are used as training data to improve the coverage and quality of two target applications (Section 3).

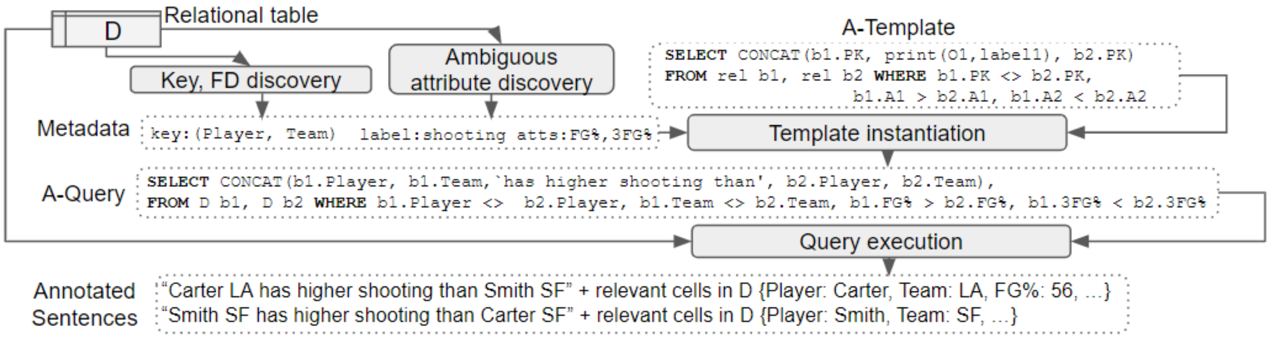

Figure 1: Pythia takes as input any relational table and generates data ambiguous sentences.

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 commit 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 1 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.

Parameter 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. This method takes as input all attribute pairs in a schema and uses a pre-trained model [3] fine-tuned with examples obtained with weak-supervision over a corpus of tables. 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 clear to a human but an algorithm detects

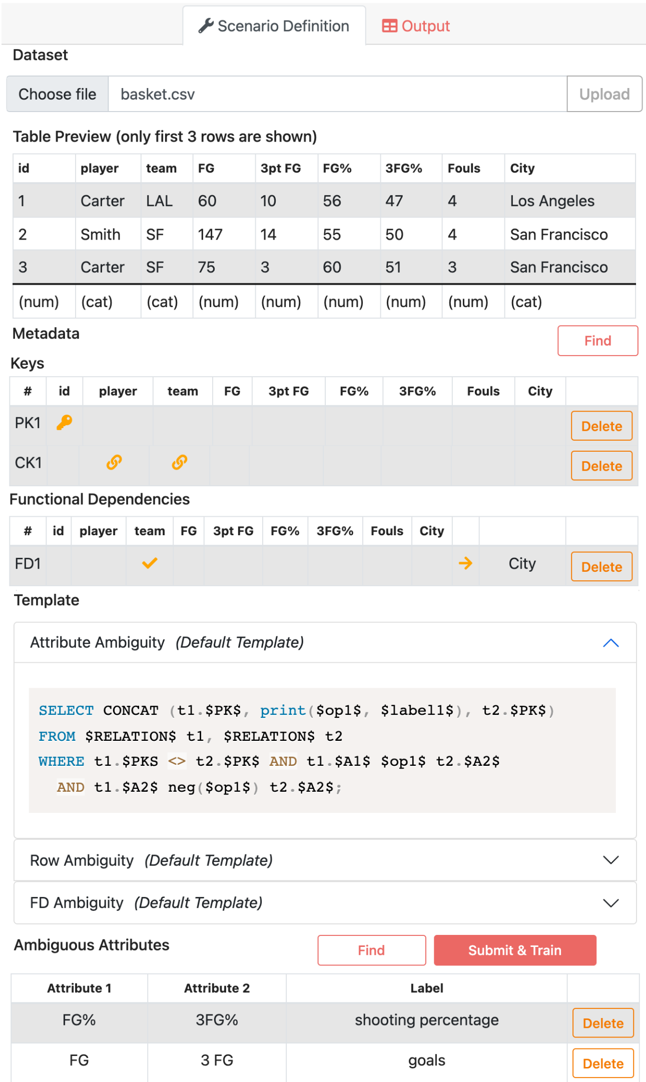

Figure 2: Pythia configuration of a scenario for a new table.

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 **fouls** and **apps**.

3 DEMONSTRATION

The demonstration will be 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

At the beginning of the demo, the visitor will be able to use preloaded scenarios or create a new one. Figure 2 shows the configuration of a new scenario. First, the visitor uploads a dataset. Then metadata such as primary keys, composite keys and FDs can be manually provided or obtained with profiling [1]. The core of

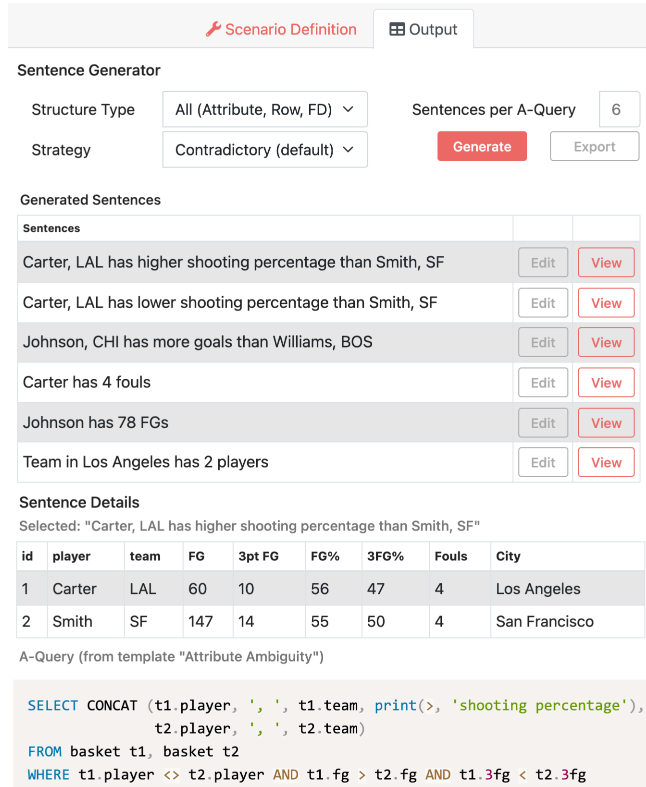

Figure 3: Output visualization with data ambiguous sentences and corresponding cells from the table.

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

After the scenario's definition, the visitor will generate ambiguous sentences as depicted in Figure 3. 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. Depending on the dataset and schema size, the number of generated sentences can vary between thousands and millions even for medium size datasets. 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

The visitor will use seven preloaded datasets. Training data for downstream applications will be generated using three datasets, the remaining datasets will be used for test data. For each dataset the visitor will 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.

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

	BLEU	ROUGE
Original (**ToTTo** [8])	0.547	0.202
ToTTo fine-tuned with PYTHIA's output	**37.631**	**0.631**

ToTTo. ToTTo [8] 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. The visitor will test a T5 model [3] trained on the original ToTTo for sentence generation. The results will show that ambiguities are not handled in ToTTo, and thus, the model simply fails when it makes predictions over P_d. The visitor will then use the P_t to fine-tune the model. Such fine-tuning step will improve the performance of the model in terms of BLEU and ROUGE metrics according to the results in Table 2 (higher is better).

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

	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

Feverous. Feverous [5] 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. We show the attendant that a model trained with the Feverous baseline pipeline 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 will show the verification of statistical claims related to COVID-19 [6]. We incorporate the contributions of PYTHIA to improve an existing system (https://coronacheck.eurecom.fr/). The original system was not able to handle attribute ambiguity. Also, row 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 same location but different timestamps, such as "In France, 10k confirmed cases have been reported" (today or yesterday?).

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

	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

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] Ziawasch Abedjan, Lukasz Golab, Felix Naumann, and Thorsten Papenbrock. 2018. *Data Profiling*. Morgan & Claypool Publishers.

[2] Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Yang Wang, and William W. Cohen. 2021. Open Question Answering over Tables and Text. In *ICLR*. OpenReview.net.

[3] Colin Raffel et al. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21, 140 (2020).

[4] Preslav Nakov et al. 2021. Automated Fact-Checking for Assisting Human Fact-Checkers. In *IJCAI*. 4551–4558.

[5] Rami Aly et al. 2021. FEVEROUS: Fact Extraction and VERification Over Unstructured and Structured information. In *Thirty-fifth Conference on Neural Information Processing Systems (NIPS - Datasets and Benchmarks)*.

[6] Georgios Karagiannis, Mohammed Saeed, Paolo Papotti, and Immanuel Trummer. 2020. Scrutinizer: A Mixed-Initiative Approach to Large-Scale, Data-Driven Claim Verification. *Proc. VLDB Endow.* 13, 11 (2020), 2508–2521.

[7] [n.d.]. 2021. Notes on Ambiguity. https://cs.nyu.edu/~davise/ai/ambiguity.html. (2021).

[8] 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. In *EMNLP*. ACL, 1173–1186.

[9] James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, and Alon Y. Levy. 2021. From Natural Language Processing to Neural Databases. *Proc. VLDB Endow.* 14, 6 (2021), 1033–1039.

[10] Immanuel Trummer. 2019. Data Vocalization with CiceroDB. In *CIDR*. www.cidrdb.org.

[11] You Wu, Pankaj K. Agarwal, Chengkai Li, Jun Yang, and Cong Yu. 2017. Computational Fact Checking through Query Perturbations. *ACM Trans. Database Syst.* 42, 1 (2017), 4:1–4:41.