PyAdel: A relationship preserving multi-table synthetic data generation tool

Saurabh Gupta
IIIT-Delhi
India
saurabhg@iiitd.ac.in

Shaurya Dewan
IIIT-Hyderabad
India
shaurya.dewan@students.iiit.ac.in

Ansh Khandelwal
IIIT-Hyderabad
India
ansh.khandelwal@students.iiit.ac.in

Sujith Joseph
Infosys
India
sujith_joseph01@infosys.com

Arun Balaji Buduru
IIIT-Delhi
India
arunb@iiitd.ac.in

Ponnurangam Kumaraguru
IIIT-Hyderabad
India
pk.guru@iiit.ac.in

ABSTRACT

Data hungry machine learning and deep learning algorithms require a lot of data to produce better results. While the data is available for generic tasks, there is scarcity when it comes to specific domains like healthcare, edge computing, etc. To solve the scarcity problem, researchers have proposed techniques to generate images, tabular data, videos, text, and so on. However, when it comes to multi-table data having relationships, the tabular data generation algorithms fail to perform well. Through this research, we create a python package, known as *pyadel*, to generate multi-table synthetic data where the relationships within and among the tables are preserved. *pyadel* automatically detects the type of attributes present in the table, detects the primary keys, finds the foreign key-primary key (FK-PK) relationships, discovers constraints among attributes, and generates synthetic data that mimics the distribution of the original dataset. The generated dataset can then be used for training models or sharing data publicly.

KEYWORDS

synthetic data generation, automatic relationship discovery, machine learning

ACM Reference Format:
Saurabh Gupta, Shaurya Dewan, Ansh Khandelwal, Sujith Joseph, Arun Balaji Buduru, and Ponnurangam Kumaraguru. 2018. *PyAdel*: A relationship preserving multi-table synthetic data generation tool . In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

With the advancements in machine learning and deep learning techniques, the learning algorithms are becoming more and more data-hungry to generate better results. Large AI models like DALL-E [13], GPT-3 [5] require humongous data to generate state-of-the-art results. Data availability is not an issue for generic tasks, but when it comes to domain-specific learning tasks, we see scarcity of data as a bottleneck in creating a solution. Healthcare and edge computing are perfect examples of domains where data is either scarce or at times not available at all [1, 14]. Another challenge that we face when dealing with data is the restrictions on the amount of data that can be released due to privacy, security and legal compliance issues [6, 8, 9]. For instance, the US Census Bureau releases only 1% of its table of records every year, along with statistics about the entire table. However, the machine learning (ML) models trained on the released sub-table are usually sub-optimal [7].

Generating synthetic data for a single table is easy as it does not involve finding primary keys, consistently making sure that the primary key - foreign key relationship is maintained, and constraints are discovered automatically. Researchers have used generative adversarial networks (GANs) to generate tabular data in the past [3, 4, 18]. None of them successfully maintain relationships and follow constraints when it comes to multi-table synthetic data generation. [17] successfully detects the type and primary key, but posits relationship discovery as a learning problem resulting in failure to detect FK-PK relationships efficiently. It is also limited when it comes to constraint discovery, as the authors only consider four hard constraints viz., less than, greater than, addition and subtraction. The authors also stated that their method does not work on the Biodegradability dataset [2]. In another work, the authors assume the schema to be available and only explore to find functional dependencies (constraints) in single table datasets using autoencoders [7]. *pyadel* overcomes the relationship discovery challenge through a deterministic algorithm and uses an empirical guide for choosing the right estimators along with autoencoders to find functional dependencies that span across multiple tables. We choose Biodegradability along with three other datasets (Carcinogenesis [15], Hepatitis [10], and Bupa [11]) to demonstrate our results. We do not share the code publicly due to IP issues, but we generate synthetic datasets and have posted them publicly at: https://bit.ly/3RGLbkI.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

2 SYSTEM DESIGN AND IMPLEMENTATION

The proposed system, *pyadel*, consists of five components: **Type Detection**, **Primary Key Discovery**, **Relationship Discovery**,

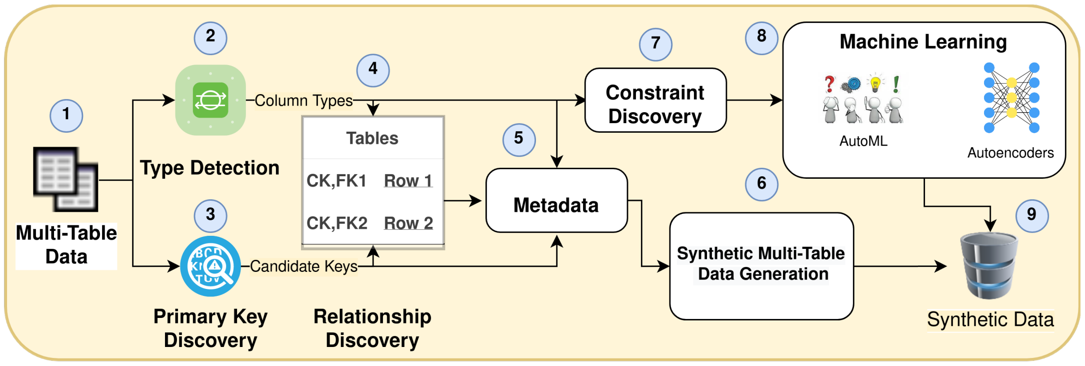

Figure 1: *pyadel* Flow Diagram: 1) We pass multiple related tables as input without the schema, 2) Detect the types of attributes of all tables, 3) Perform primary key discovery, 4) Perform relationship discovery, 5) Use types, primary keys and relations to generate metadata in the required format, 6) Use metadata to generate multi-table data from SDV, 7) Find constraints, 8) Filter and model constraints using machine learning, 9) Finalize synthetic data.

Constraint Discovery, and **Data Generation** as shown in Figure 1.

2.1 Type Detection

We categorize attributes into higher-level types that have some semantic and contextual meaning. These types convey information about what a value represents. We consider the following types for the process of type detection: id (primary, non-primary), categorical (boolean, categorical, ordered), number (integer, float), and datetime.

The problem can be stated as: Given a set of fields $f_1 ... f_k$, data corresponding to the fields $d_1 ... d_k$, and annotations about their higher level type $type_1 ... type_k$, can we learn a model, TD such that $type_i \leftarrow TD(f_i, d_i)$.

The model, $TD(f_i, d_i)$ can then be used to predict the types of attributes in datasets where the schema is not known. We follow the following steps to detect types:

(1) Choose datasets with available data types to use as annotations. This gives us, $f_1 ... f_k$, $d_1 ... d_k$ and their types $type_1 ... type_k$.
(2) Extract features like whether the type is a string or integer, whether it is a date, or "_id" is present, etc., (all features are listed below) for all attributes $d_1 ... d_k$.
(3) Given the features, train a decision tree with the computed features and model the problem as a multi-class classification problem.
(4) Validate the classifier by predicting the data types and matching them with ground truth.
(5) Use the classifier to predict the types of data attributes in the input tables with unknown schema.

We use existing datasets, banking[1] and adult[2] to create the following features as independent variables and their types as the dependent variable: i) ratio of number of unique values to the total number of values, ii) if the number of unique values is less than a hard-coded categorical threshold, iii) if the values are integers or floating type, iv) if the values are strings, v) if the field name contains "_id" or field name is "id", vi) if the field name contains "date" or "time".

We train a simple decision tree classifier such that for new datasets we will be able to detect the types faster by computing the aforementioned features beforehand.

2.2 Primary Key Discovery

Next, we detect the primary key in each table of the dataset. The primary key in a dataset has two properties: it has a unique value for each row, and it can either be just one column (common case), or it can be a combination of attributes in a table. During automatic detection, we find the minimum set of attribute(s) that can uniquely identify each row.

The primary key discovery problem can be formulated as: Given a table with attributes $f_1 ... f_k$, find a field f_i or a combination of fields $f_i ... f_r$ where $i ... r$ is a subset of $1 ... k$, that uniquely identifies each row in the table.

Algorithm 1 performs the following steps to determine the primary key of the input table: i) identifies the candidate set, filtering the candidates. For identification, we check each attribute's feature

Algorithm 1 Primary Key Discovery

```
Require: Table $t$
Require: FILTERCANDIDATES($candidate\_sets$)   ▷ Definition as
    discussed in the text above
 1: procedure GETCANDIDATES($t$)
 2:     $possible\_cands$ ← all length-1 and length-2 sets of field
    combinations
 3:     $candidate\_sets$ ← []
 4:     for candidate_set ∈ possible_cands do
 5:         if $candidate\_set$ is unique across all rows then then
 6:             $candidate\_sets$ ← candidate_sets ∪ {candidate_set}
 7:         end if
 8:     end for
 9:     return FILTERCANDIDATES($candidate\_sets$)
10: end procedure
```

[1]https://archive.ics.uci.edu/ml/datasets/bank+marketing
[2]https://archive.ics.uci.edu/ml/datasets/adult

Algorithm 2 Relationship Discovery Algorithm

```
Require: All tables in the dataset, $tables$ and threshold, $th$
Require: type($attribute$)                                  ▷ returns the high level type of input attribute
Ensure: $relations \leftarrow$ list of relations in the format $\{child\_table, child, parent\_table, parent\}$
procedure GETRELATIONS($tables$, $th$)
    $relations \leftarrow []$
    for $child\_table \in tables$ do
        $possible\_children \leftarrow$ all attributes in $child\_table$
        for $child \in possible\_children$ do
            for $parent\_table \in tables$ do
                if $child\_table \neq parent\_table$ then
                    $possible\_parents \leftarrow$ all length-1 candidate keys in $parent\_table$      ▷ from primary key discovery
                    for $parent \in possible\_parents$ do
                        $child\_only \leftarrow$ values present in child but not in parent
                        if type($parent$) == type($child$) and $count(child\_only)/count(child) < (th * count(child))$ then
                            $relations \leftarrow relations \cup \{child\_table, child, parent\_table, parent\}$
                        end if
                    end for
                end if
            end for
        end for
    end for
    return $relations$
end procedure
```

- "the ratio of the number of unique values in the dataset and the total number of rows". If the ratio is equal to 1, then the attribute is a candidate key, and in turn, is a potential primary key. To choose the minimal candidate key as the primary key, if an attribute is a candidate key, we do not consider it in the multi-attribute candidate keys.

Once we have all the candidate keys, the goal is to identify the most likely primary key from the candidate key set. Now, fields like datetime can fall in the candidate key set but might not necessarily be a primary key. Therefore, we filter out candidate keys based on their type and the fact whether the keyword "id" is present in the attribute name in any form and prefer the most minimal candidate key. For example, if we have $user\_id$ and $(user\_id, datetime)$ as candidate keys, we pick $user\_id$ as the primary key. Note that - if we get more than one candidate keys even after applying the filter, we randomly pick the first one in the list as it seems to work well as observed empirically.

2.3 Relationship Discovery

Once we detect type of each attribute in the tables, and primary keys are identified, we can perform relationship discovery. We identify the foreign key-primary key (FK-PK) relationships, which are many-to-one where entries in one table reference to the primary key of another table. All valid foreign keys contain an inclusion dependency, which states that the set of values in the foreign key attribute is a subset of the set of values in the referenced primary key attribute.

The relationship discovery problem can be formulated as: given a set of tables, $table_1, table_2, ...table_k$ and fields associated with each, $f^1_{1...k} ... f^p_{1...k}$, identify all pairs of (f^i_j, f^l_m) where $f^i_j \in f^l_m$.

Algorithm 2 defines the methodology that we follow for foreign key discovery. We first set a "threshold" on how many values we can allow to be in the foreign key attribute (of the child table) that are not originally present in the primary key attribute (of the parent table). The threshold is usually set to 0.01, i.e, 1% but can be varied if required. For our experiments, we use 0.01 as the threshold value because we want minimal FK-PK violations. The algorithm returns the relationships available in the tables in the child_table, child, parent_table, parent format, which is directly used in our data generation process.

2.4 Constraint Discovery

While generating multi-table data, even when the schema is available there are constraints within the data that are not explicitly defined in the schema. For example, in a table having the fields start-date and end-date, start-date will always have an earlier date value as compared to end-date. While generating such multi-table data, we need to consider such constraints as well.

In order to do it automatically, we will require to explore all possibilities in the given tables. If we have t tables and n variables in each table, the total number of variables will be $n \times t$. Now, among the $n \times t$ variables, we need to explore the power set of the $n \times t$ variables, and the number of possible models to train will be $2^{(n \times t)}$. For a simple dataset with just 20 variables spread across multiple tables, we will require to train $2^{(20-(empty\_sets, id\_attributes))}$ approximately equal to 1 million models, given we do not consider the empty sets and the id columns as they do not add any meaning. Empirically, we find that training one simplistic linear regression model with just 50 samples takes 53 seconds on a 2.30GHz Intel Xeon processor with four CPUs. For 1 million models, it will take 600 days or 1.5 years.

To bring down the exponential possibilities of constraints, we added a user intervention step, where we externally capture the independent and dependent variables among which constraint discovery is required. Based on the domain, once we know the dependent and independent variables, we use machine learning (roadmap[3] and autoencoders [20]) to find the constraints between the variables.

2.5 Private Synthetic Tabular Data Generation

For synthetic data generation, we use the synthetic data vault (SDV) [12] as the framework.[4] SDV allows users to learn single-table, multi-table and time-series datasets to later on generate new synthetic data that has the same format and statistical properties as the original dataset. SDV uses conditional GAN [18], vine copula models [16], tabular GAN [19], etc., to generate single table synthetic data. For multi-table data, SDV uses a hierarchical modeling algorithm that allows to recursively walk through a relational dataset and apply tabular models across all the tables in a way that lets the models learn how all the fields from all the tables are related [12].

3 DEMONSTRATION

We perform experiments with relational datasets taken from the relational dataset repository.[5] To demonstrate *pyadel*, we show the results on four datasets: Biodegradability, Carcinogenesis, Hepatitis, and Bupa. All four of them are healthcare datasets where data is scarce and its availability is a problem. Table 2 shows that *pyadel* works perfectly with all datasets, correctly finds the primary keys, and does not miss any FK-PK relationship while running automatically without a readily available schema.

[3]https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html
[4]https://sdv.dev/SDV/index.html
[5]https://relational.fit.cvut.cz

We further use the Biodegradability [2] dataset to demonstrate primary key and relationship discovery along with effective synthetic data generation. It is a dataset of chemical structures containing 328 compounds labeled by their half-life for aerobic aqueous biodegradation. It has five tables: molecule, atom, bond, gmember, and group with five FK-PK relationships among these tables in the original dataset. Figure 2 shows that *pyadel* successfully finds the types, primary keys, and the FK-PK relationships between the tables without passing any database schema as input. In Table 1, we show that the real and the synthetic data have closer statistical values and are similarly distributed.

4 CONCLUSION

In this work, we present *pyadel*, a python package to generate multi-table synthetic data that preserves the relationships within and among the tables. *pyadel* automatically detects the type of attributes present in the tables, detects the primary keys, finds the foreign key-primary key (FK-PK) relationships, discovers constraints among attributes, and generates synthetic data that mimics the distribution of the original dataset. The generated dataset can then be used for training models or sharing data publicly.

Dataset	PK Violations	Missed FK-PK relationships
Biodegradability	0/5	0/5
Carcinogenesis	0/6	0/3
Hepatitis	0/7	0/4
Bupa	0/9	0/2

Table 2: Primary key violations and missed FK-PK relationships. *pyadel* perfectly works with all four datasets.

Attribute	Mean	Std	Min	25%	50%	75%	Max
activity_real	6.649990	1.570540	2.14007	5.886100	6.040250	7.822440	11.274200
activity_synth	6.609475	1.553736	2.14007	5.532201	6.617384	7.685418	11.274200
logp_real	2.287866	2.201955	-3.81000	0.740000	2.075000	3.517500	12.110000
logp_synth	2.296670	2.170076	-3.81000	0.817500	2.420000	3.762500	9.180000
mweight_real	177.712790	107.968022	27.02590	102.154000	154.211000	236.869250	1019.640000
mweight_synth	180.908289	98.559722	27.02590	105.667057	179.818606	250.879095	498.472483

Table 1: Statistical comparison of real and synthetic dataset distribution. We see no significant differences in values representing similarity in real and synthetic dataset.

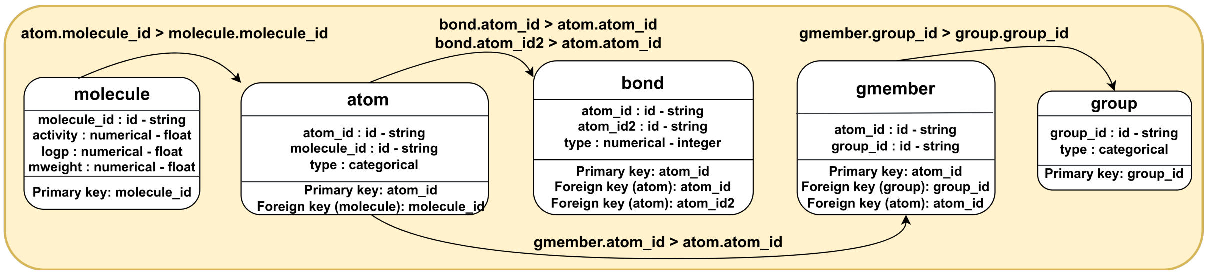
atom.molecule_id > molecule.molecule_id
bond.atom_id > atom.atom_id
bond.atom_id2 > atom.atom_id
gmember.group_id > group.group_id
molecule
molecule_id : id - string
activity : numerical - float
logp : numerical - float
mweight : numerical - float
Primary key: molecule_id
atom
atom_id : id - string
molecule_id : id - string
type : categorical
Primary key: atom_id
Foreign key (molecule): molecule_id
bond
atom_id : id - string
atom_id2 : id - string
type : numerical - integer
Primary key: atom_id
Foreign key (atom): atom_id
Foreign key (atom): atom_id2
gmember
atom_id : id - string
group_id : id - string
Primary key: atom_id
Foreign key (group): group_id
Foreign key (atom): atom_id
group
group_id : id - string
type : categorical
Primary key: group_id
gmember.atom_id > atom.atom_id

Figure 2: Demonstration of *pyadel* on Biodegradability dataset. We successfully find the correct primary keys and foreign key-primary key relationships.

REFERENCES

[1] Ms Aayushi Bansal, Dr Rewa Sharma, and Dr Mamta Kathuria. 2021. A Systematic Review on Data Scarcity Problem in Deep Learning: Solution and Applications. *ACM Computing Surveys (CSUR)* (2021).

[2] Hendrik Blockeel, Sašo Džeroski, Boris Kompare, Stefan Kramer, and Bernhard Pfahringer. 2004. Experiments In Predicting Biodegradability. *Applied Artificial Intelligence* 18, 2 (2004), 157–181. https://doi.org/10.1.1.2.3797

[3] Stavroula Bourou, Andreas El Saer, Terpsichori-Helen Velivassaki, Artemis Voulkidis, and Theodore Zahariadis. 2021. A Review of Tabular Data Synthesis Using GANs on an IDS Dataset. *Information* 12, 9 (2021). https://doi.org/10.3390/info12090375

[4] Bauke Brenninkmeijer, A de Vries, E Marchiori, and Youri Hille. 2019. *On the generation and evaluation of tabular data using GANs.* Ph. D. Dissertation. Radboud University Nijmegen, The Netherlands.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[6] Declan Butler. 2007. Data sharing threatens privacy. *Nature* 449, 7163 (2007), 644–646.

[7] Haipeng Chen, Sushil Jajodia, Jing Liu, Noseong Park, Vadim Sokolov, and VS Subrahmanian. 2019. FakeTables: Using GANs to Generate Functional Dependency Preserving Tables with Bounded Real Data.. In *IJCAI*. 2074–2080.

[8] Saurabh Gupta, Agarwal Anant, Suryatej Reddy Vyalla, Arun Balaji Buduru, and Ponnurangam Kumaraguru. 2020. Ivoted to# igotpwned: Studying voter privacy leaks in indian lok sabha elections on twitter.(2020). *Saurabh Gupta* (2020).

[9] Srishti Gupta and Ponnurangam Kumaraguru. 2013. OCEAN: Open-source Collation of eGovernment data And Networks-Understanding Privacy Leaks in Open Government Data. *arXiv preprint arXiv:1312.2784* (2013).

[10] Hassan Khosravi, Oliver Schulte, Jianfeng Hu, and Tianxiang Gao. 2012. Learning compact Markov logic networks with decision trees. *Machine Learning* 89, 3 (2012), 257–277. https://doi.org/10.1007/s10994-012-5307-6

[11] James McDermott and Richard S Forsyth. 2016. Diagnosing a disorder in a classification benchmark. *Pattern Recognition Letters* 73 (2016), 41–43.

[12] Neha Patki. 2016. *The synthetic data vault: generative modeling for relational databases.* Ph. D. Dissertation. Massachusetts Institute of Technology.

[13] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-Shot Text-to-Image Generation. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8821–8831. https://proceedings.mlr.press/v139/ramesh21a.html

[14] Sejin Seo, Seung-Woo Ko, Sujin Kook, and Seong-Lyun Kim. 2022. Understanding uncertainty of edge computing: New principle and design approach. In *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*. IEEE, 1–7.

[15] Ashwin Srinivasan, Ross Donald King, S. H Muggleton, and M.J.E. Sternberg. 1997. Carcinogenesis predictions using ILP. *Inductive Logic Programming* 1297 (1997), 273–287. https://doi.org/10.1007/3540635149_56

[16] Yi Sun, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Learning vine copula models for synthetic data generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 5049–5057.

[17] Katharine Xiao. 2017. *Towards automatically linking data elements.* Ph. D. Dissertation. Massachusetts Institute of Technology.

[18] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling tabular data using conditional gan. *Advances in Neural Information Processing Systems* 32 (2019).

[19] Lei Xu and Kalyan Veeramachaneni. 2018. Synthesizing tabular data using generative adversarial networks. *arXiv preprint arXiv:1811.11264* (2018).

[20] Junhai Zhai, Sufang Zhang, Junfen Chen, and Qiang He. 2018. Autoencoder and its various variants. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 415–419.