CAPE: Explaining Outliers by Counterbalancing

Zhengjie Miao[*1], Qitian Zeng[*2], Chenjie Li[2],
Boris Glavic[2], Oliver Kennedy[3], Sudeepa Roy[1]

[1]Duke University, Durham, NC [2]IIT, Chicago, IL [3]SUNY Buffalo, Buffalo, NY

{zjmiao,sudeepa}@cs.duke.edu, {qzeng3,cli112,bglavic}@{hawk}.iit.edu,
okennedy@buffalo.edu

ABSTRACT

In this demonstration we showcase CAPE, a system that explains surprising aggregation outcomes. In contrast to previous work, which relies exclusively on provenance, CAPE explains outliers in aggregation queries through related outliers in the opposite direction that provide *counterbalance*. The foundation of our approach are *aggregate regression patterns (ARPs)* that describe coarse-grained trends in the data. We define outliers as deviations from such patterns and present an efficient algorithm to find counterbalances explaining outliers. In the demonstration, the audience can run aggregation queries over real world datasets, identify outliers of interest in the result of such queries, and browse the patterns and explanations returned by CAPE.

PVLDB Reference Format:
Zhengjie Miao, Qitian Zeng, Chenjie Li, Boris Glavic, Oliver Kennedy, and Sudeepa Roy. CAPE: Explaining Outliers by Counterbalancing. *PVLDB*, 12(12): xxxx-yyyy, 2019.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

1. INTRODUCTION

When analyzing a dataset by running aggregation queries, a user may encounter a result that is higher or lower than her expectation. Given such an outlier, the user may want to know what led to this unexpected outcome. For instance, consider a publication dataset such as the one shown in Table 1. User Alice analyses trends in the publication output of researchers in Computer Science. Inspecting the result of an aggregation query that returns the total number of publications per author and year, she observes that a particular author published significantly more papers in 2018 than usual. Previous approaches [3, 4] explain outliers in aggregation query results through *intervention*, i.e., they compactly encode a part of the provenance [1] for the result of interest whose exclusion from the input would significantly change the aggregation result in the opposite direction of the outlier of interest. For instance, they may explain the author's high publication count in 2018 by listing the venues where she published most frequently in this particular year.

Pub

author	pubid	year	venue
A_X	P1	2005	SIGKDD
A_X	P2	2004	SIGKDD
A_Y	P2	2004	SIGKDD
A_Z	P3	2004	SIGMOD

Table 1: Running example publications dataset

While the techniques from [3, 4] produce meaningful explanations for some scenarios, these techniques have limited applicability when the 'cause' of an outlier is not contained in its provenance, or when we want to explain a low outlier in the output of a monotone aggregate query (excluding inputs will only further decrease the aggregate). As an example, a possible explanation for someone's high publication count in 2018 might be that most of her submissions in 2017 got rejected, but resubmissions of these papers got accepted in 2018 in addition to other publications based on new work. That is, a higher than expected value (the number of publications in 2018) is explained by a lower than expected value for another answer tuple (the number of publications in 2017). We refer this type of explanation where an outlier in one direction is explained by a *related* outlier in the opposite direction as *explanation by counterbalance*. In this demonstration, we showcase CAPE **(Counterbalancing with Aggregation Patterns for Explanations)**, a system we have developed for computing such explanations. For additional technical details we refer the reader to [2].

Since counterbalances are also outliers (values that deviate from an *expected* value), we need to determine what values to expect in the result of an aggregation query. To this end, we define *aggregate regression patterns* (ARPs), which model correlations in the result of a group-by query. In [2] we present an efficient method for mining such patterns over a given dataset. Furthermore, we introduce an algorithm for finding counterbalances that are related (*"similar"*) to the outlier of interest and deviate significantly (*"outlierness"*) from the expectation determined by a pattern. CAPE ranks possible explanations based on a scoring function that combines similarity and outlierness.

The goal of this demonstration is to showcase the effectiveness and efficiency of CAPE. Cape is available as open source at `https://github.com/capeexplain/cape`.We will first introduce ARPs and explanations by counterbalance, and discuss the design and implementation of CAPE. Afterwards, we present real-world demonstration scenarios and explain CAPE's user interface. A detailed discussion of related work can be found in [2].

[*]denotes equal contribution.

Proceedings of the VLDB Endowment, Vol. 12, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/xxxxxxx.xxxxxxx

author	venue	year	pubcnt
A_X	SIGKDD	2006	4
A_X	SIGKDD	2007	1
A_X	SIGKDD	2008	4
A_X	VLDB	2006	4
A_X	VLDB	2007	4
A_X	VLDB	2008	1
A_X	ICDE	2006	6
A_X	ICDE	2007	6
A_X	ICDE	2008	4

Table 2: Partial result of query Q_0 from Example 1

Rank	Explanation	score
1	(A_X, ICDE, 2007, 6)	13.78
2	(A_X, ICDE, 2006, 6)	10.91
3	(A_X, ICDM, 2007, 5)	6.44
4	(A_X, VLDB, 2007, 4)	5.77
5	(A_X, SIGMOD, 2008, 4)	5.57

Table 3: Top-5 explanations returned by CAPE for Ex. 1

2. BACKGROUND AND CAPE OVERVIEW

We now briefly introduce user questions, aggregate regression patterns (ARPs), and explanations by counterbalance. We use the example shown below as a running example.

EXAMPLE 1. *Consider a simplified and anonymized version of the DBLP (*`http://dblp.uni-trier.de/`*) publication dataset with schema* `Pub(author,pubid,year,venue)`*. Some example entries are shown in Table 1. The following query Q_0 computes the number of publications per author, year, and venue. Table 2 shows a subset of this query's result.*

```
SELECT author, year, venue, count(*) AS pubcnt
FROM Pub
GROUP BY author, year, venue
```

Given this result, a user may wonder why certain rows have a higher/lower aggregation function value than expected. For instance, if the user is aware that author A_X is a prolific data mining researcher who typically publishes multiple papers in SIGKDD each year, she may wonder "why did A_X publish only 1 paper in SIGKDD in 2007?".

User question. Consider a relation R and a group-by aggregate query $Q = \gamma_{G,agg(A)}(R)$, where G denotes the query's group-by attributes, A is an attribute from R, and agg is an aggregation function (*sum* or *count*). Given a result tuple $t \in Q(R)$ and a direction $dir \in \{high, low\}$, a *user question* is a quadruple: $\phi = (Q, R, t, dir)$.

Intuitively, while exploring the data by running query Q over R, the user observes that the aggregation value of an answer tuple t is lower or higher than her expectation and wants to understand why this is the case. For instance, the question given in Example 1 can formally be written as $\phi_0 = (Q_0, Pub, t_0, low)$, where $Q_0 = \gamma_{\texttt{author,year,count(pubid)}}(Pub)$ and t_0 is the tuple $(A_X, SIGKDD, 2007, 1)$ from Table 2.

2.1 Aggregate Regression Patterns

Aggregate regression patterns (ARPs) model trends that hold in aggregation. For the DBLP dataset we may find ARPs like P_1 (where 'many' depends on local and global thresholds δ, θ, Δ, λ described below):

P_1 := *"for many authors, the number of publications increases linearly over the years".*

Pattern P_1 states that if we partition the rows in the output of the query $Q_0 = \gamma_{\texttt{author,year,count(pubid)}}(Pub)$ based on their `author` names, then within each fragment the relationship between the `year` values and the aggregate output `count(pubid)` (number of publications) can be described approximately though a linear function.

An ARP partitions the result of an aggregation query Q on a subset $F \subseteq G$ of the group-by attributes (the **partition attributes**), and within each fragment describes correlations between the remaining group-by attributes $V = (G - F)$ (the **predictor attributes**) and the results of the aggregation function using a regression model. Pattern P_1 uses a single partition attribute $F = \{\texttt{author}\}$ and single predictor attribute $V = \{\texttt{year}\}$. In addition to the partition and predictor attributes, an ARP also specifies an aggregate function and attribute to be used (`count`(*pubid*) in P_1) and what regression model type to use: e.g., *linear*.

Given an ARP P, we want to check whether P correctly describes a trend in the data. We consider a pattern to hold **"locally"** (for a given fragment) if there are sufficient distinct values of V within the fragment (larger than a threshold δ which we refer to as the *local support threshold*), and we can train a regression model on this fragment with a sufficiently high *goodness-of-fit* (higher then a *local model quality threshold* θ). For instance, P_1 holds locally for author A_X if she has actively published in more than δ years and we can fit a linear regression that predicts her number of publications based on the year with a goodness-of-fit that is higher than θ. A pattern is considered to hold **"globally"** if the percentage of fragments for which it holds locally exceeds an adjustable threshold Δ (the *global support threshold*) and the ratio between the number of fragments with sufficient evidence (local support) for which the pattern holds is larger than a threshold λ (the *global confidence threshold*). For example, P_1 would be considered to hold globally if it holds for more than Δ authors and out of all authors that published in more than δ years, the pattern holds for a fraction that is larger than λ (details in [2]).

2.2 Explanations by Counterbalance

Using ARPs, CAPE finds explanations by *counterbalance* as introduced in the introduction. Intuitively, an explanation comes from a fragment f' generated by fixing values for the partition attributes F' of a pattern P' that can be applied to the user question. That is, f' agrees with the user question tuple t on attributes $F' \cap G$ where G are the group-by attributes of the user's query. A counterbalance is a data point from fragment f' that is an outlier in the opposite direction of the user question, i.e., it is lower/higher then the prediction according to pattern P'. The actual details are more involved; and we refer the reader to [2].

The top-5 explanations produced by CAPE for ϕ_0 from Example 1 are shown in Table 3. The **score of an explanation** is computed as $\frac{dev \times isLow}{distance \times Norm}$, where (i) *dev* denotes the relative deviation of the counterbalance from the value predicted for this data point by the pattern (e.g., a higher count of ICDE publications for A_X is more surprising than a higher count of ICDM papers as A_X is a data mining researcher and ICDM is a data mining conference), (ii) *distance* denotes the distance between the group-by attributes of the user question tuple t and the values of the counterbalance (e.g., explanations from the same year 2007 or adjacent years 2006/2008 have higher scores), (iii) $Norm$ is a normalization factor, and (iv) $isLow$ is a flag that records whether $dir = low$ in the user question. The top two explanations for the running example (ICDE 2007 and ICDE 2006) suggest that A_X might have sent more pa-

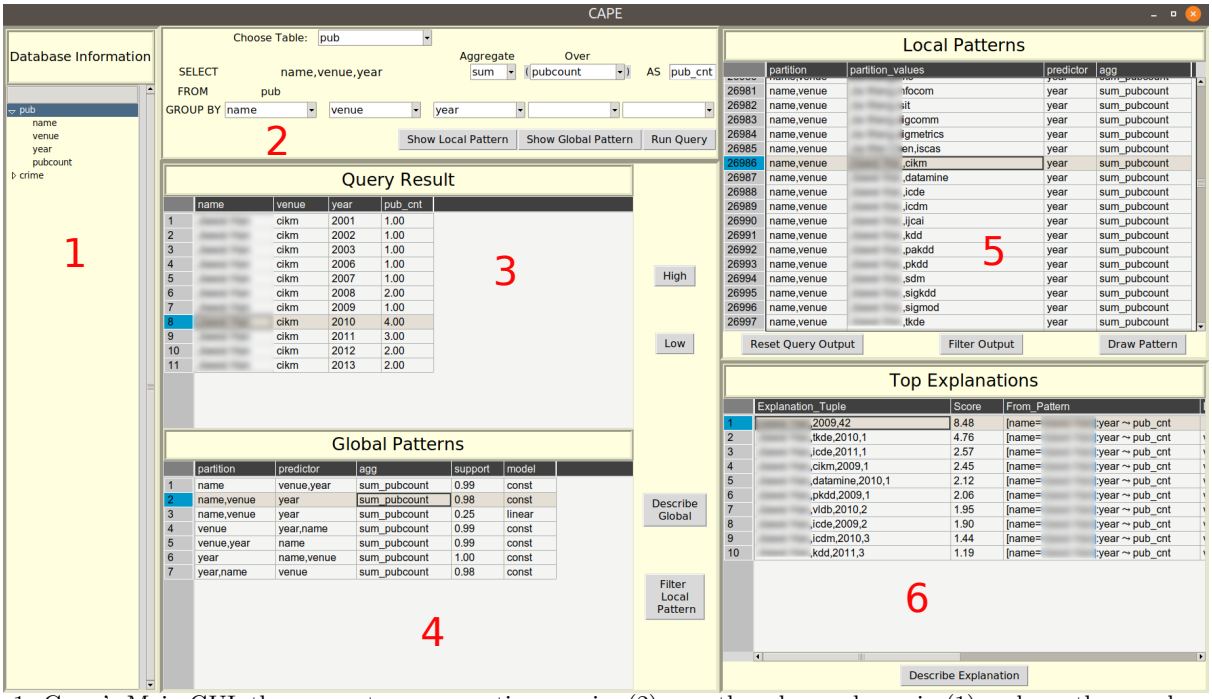

Figure 1: CAPE's Main GUI: the user enters aggregation queries (2) over the schema shown in (1) and can then explore query results (3), browse patterns relevant to the query (4 + 5), and can request explanations for why an answer is high/low (6).

pers to ICDE in these years instead of to SIGKDD. None of the explanations appearing in Table 3 can be produced by previous approaches [4, 3] since these explanations do not stem from the tuples in the provenance of A_X's SIGKDD publications in 2007.

2.3 Implementation and Optimizations

CAPE is implemented in Python (version 3.5) and runs on top of PostgreSQL (version 10.4). CAPE consists of two main components: (A) an offline ARP mining method, and (B) a method to compute top-k explanations interactively.

(A) Offline ARP Mining. To determine which patterns hold for a dataset we have to enumerate all possible ARP candidates, and for each candidate determine whether it holds locally for a sufficiently large number of fragments. Since the number of possible pattern candidates is exponential in the number of attributes of the input relation, testing all pattern candidates using a brute force algorithm is not efficient. The algorithm we presented in [2] scales to larger datasets by applying several optimizations. To determine whether a particular pattern candidate $P = (F, V, agg, A, M)$ (with partition attributes F, predictor attributes V, aggregate function $agg(A)$, and regression model M) holds locally over R, we need run a query $Q(R) = \gamma_{F \cup V, \mathtt{agg}(A)}(R)$ to retrieve the relevant data and then train a regression model. **(1) Reuse queries:** We reduce the query time by using a single query for all pattern candidates that share the same $F \cup V$ – this query retrieves all fragments for these global pattern candidates. For instance, if $F_1 = \{A, B, C\}$ and $F_2 = \{A, C\}$ and $V_1 = \{D\}$ and $V_2 = \{B, D\}$ a single querying grouping on $\{A, B, C, D\}$ is sufficient to test both patterns. **(2) Limit the number of candidate ARPs:** We only consider patterns for which $|F \cup V| \leq \psi$ for a configurable threshold ψ since the number of group-by attributes $|G|$ in a user's query is typically small and we are unlikely to use patterns where $|F \cup V|$ is much larger than $|G|$. **(3) Use FDs:** We also detect functional dependencies (FDs) during pattern mining and use these FDs to avoid mining redundant patterns, e.g., consider P_1 and P_2 that differ only in F, say $F_1 = \{A\}$ and $F_2 = \{A, B\}$ and there is an FD $A \rightarrow B$. If P_1 holds, then P_2 will also hold, i.e., we can avoid checking whether P_2 holds.

(B) Finding the top-k explanations. Using ARPs that are mined offline, CAPE finds explanations to a given user question interactively. Two optimizations are used. **(1) Min-heap:** We use a min-heap-based algorithm for generating and ranking explanations. The algorithm takes as input a set of ARPs $\mathcal{P}$ and a user question $\phi = (Q, R, t, dir)$. The min-heap contains the top-k explanations with highest scores found so far. The algorithm iterates over all patterns in $\mathcal{P}$ and all their refinements. If it finds a relevant pattern and a new candidate explanation, then it checks whether the explanation's score is higher than the score of the root element of the min-heap; if so, the root element is replaced by the new explanation and the heap is updated. **(2) Use an upper bound on scores for an ARP:** For each ARP P in $\mathcal{P}$ we compute an upper bound on the score of any explanation for the user question that uses P. We skip any candidate explanation whose upper bound is lower than the lowest score of the current top-k explanations.

3. DEMONSTRATION

Datasets. We will use three real world datasets in the demonstration of CAPE. In addition to the *DBLP* dataset described in Example 1, the user can also explore CAPE using a Crime dataset from the City of Chicago data portal and a NYC Taxi dataset [2]. The main user interface of CAPE is shown in Figure 1, where the user can interact with CAPE as follows.

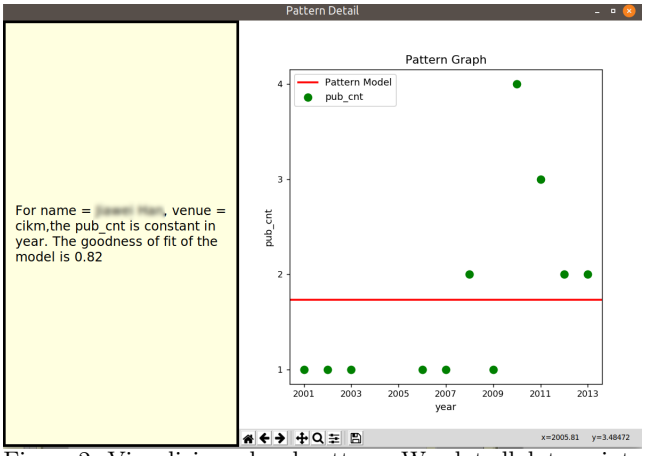

Figure 2: Visualizing a local pattern. We plot all data points as well as the regression model.

1. Run aggregation queries and inspect global and local ARPs. To help the user to formulate queries, CAPE shows the schema of the underlying database in the left panel ①. The user enters aggregation queries into box ②. Query results are shown in ③. The user can select any tuple and use the "High" and "Low" buttons to request the system to explain why the currently selected tuple's aggregation value is higher or lower than expected. To aid the user in finding interesting results, the query result can be re-sorted on any of the result columns. When a query is run, based on the group-by attributes CAPE determines the ARPs that are relevant to the query. Global patterns are shown in ④, and once a global pattern is clicked, the corresponding local patterns are shown in ⑤.

2. Inspecting global and local patterns. The user can request a more detailed explanation for a global pattern by clicking the "Describe Global" button. For local patterns the "Draw Local" button opens up a pop-up window which shows additional information about the pattern and a plot showing the corresponding data points as well as the prediction by the model based on which the pattern holds. Figure 2 shows such a plot for the pattern P_2: *"for many (author, venue) pairs, the number of publications is constant over the years"* (for partition attributes values: *author* = A_X, and *venue* = CIKM). The regression model trained on the data points shown in the figure predicts that author A_X publishes roughly 1.73 papers every year in CIKM. The plots for local patterns are useful for understanding which query answers are outliers, and can aid the user in determining what questions to ask.

3. Use filters to explore patterns and query results. CAPE further helps a user find interesting questions by filtering out patterns and query answers. The user can select any number of global patterns for which the local patterns are then shown (④). They can also select one or more local patterns to filter the query result so that only tuples that contributed to the selected patterns will be shown (⑤).

4. Generating and visualizing explanations. The user can select any tuple t in the query result with or without CAPE's help and click on the "*High*" or "*Low*" buttons to create a user question $\phi = (Q, R, t, high/low)$. CAPE will generate explanations online and return the top-k explanations (default is $k = 10$) ranked by score. Each explanation consists of a pattern P that is relevant to ϕ, a pattern P' that

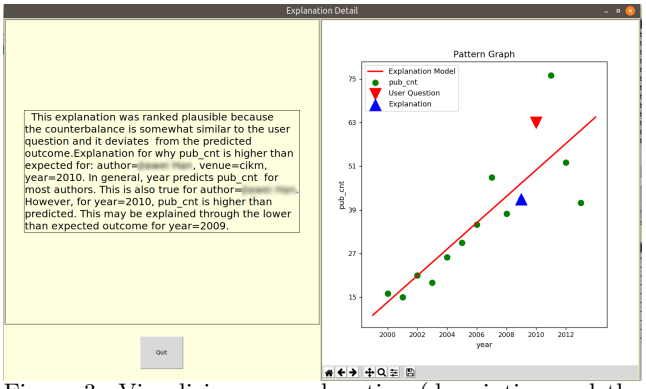

Figure 3: Visualizing an explanation (description and the relevant pattern). The tuple from the user question is the (bigger) red downward triangle and the counterbalancing explanation is the blue upward triangle.

is a 'refinement' of P (details in [2]), and a counterbalance tuple t' which is an outlier according to P' in the opposite direction of t in ϕ. The top-k explanations are shown in box ⑥. Clicking on an explanation opens a pop-up window, which shows a textual description of the explanation and one or two plots for the ARPs P and P' (only one plot is shown if $P = P'$). Furthermore, the data point corresponding to the user question and the data point corresponding to the counterbalance are highlighted (red downward triangle and blue upward triangle). Figure 3 shows an explanation for user question "*Why is the number of A_X's CIKM publications in 2010 so high?*". Here the relevant pattern P and refined pattern P' are the same: P_3 = *"for many authors, the number of publications increases linearly with years"*. P_3 holds holds locally for A_X. The counterbalance comes from his total number of publications in 2009, which is lower than the predicted value. This explanation can be verbally described as "*A_X's number of publication at CIKM in 2010 is high, possibly his publications in 2009 is less than expected*". One possible interpretation is that some of his papers not published in 2009 ended up being published in 2010.

Note that, like previous work, the explanations returned by CAPE *can guide the user in understanding query results. Obviously, not all outliers can be explained through counterbalancing, e.g., an outlier may be by a data entry error. Integrating our ideas with other techniques for explaining outliers is an interesting and challenging research direction.*

Acknowledgments. Supported in part by NSF Awards IIS-1552538, IIS-1703431, IIS-1750460, OAC-1541450, OAC-1640864, SMA-1637155, and NIH Award 1R01EB025021-01.

4. REFERENCES

[1] T. J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.

[2] Z. Miao, Q. Zeng, B. Glavic, and S. Roy. Going beyond provenance: Explaining query answers with pattern-based counterbalances. In *SIGMOD*, 2019.

[3] S. Roy and D. Suciu. A formal approach to finding explanations for database queries. In *SIGMOD*, pages 1579–1590, 2014.

[4] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. *PVLDB*, 6(8):553–564, 2013.