Context-Augmented Impact Summary with Attribute Filtering for Provenance Summarization

Hongtao Song
Harbin Engineering University
songhongtao@hrbeu.edu.cn

Mingxuan Liu
Harbin Engineering University
mingxuanliu@hrbeu.edu.cn

Qilong Han*
Harbin Engineering University
hanqilong@hrbeu.edu.cn

Zhiwen Yu
Harbin Engineering University
zhiwenyu@hrbeu.edu.cn

ABSTRACT

As user demands for data quality increase, scholars are increasingly focusing on data provenance-based query explanation techniques. Provenance summarization has proven beneficial for explaining aggregate query results in relational databases, providing multiple feature rules to explain user questions. However, existing provenance summarization methods, while effective in summarizing user question provenance, suffer from limited information and poor interpretability in generated summaries. One main issue is that they offer users top-k feature rules without distinguishing between public features from the entire dataset and private features specific to the user question. When users without prior knowledge utilize the current summarization algorithm to generate top-k summaries, it becomes challenging to accurately locate answers to the questions. Thus, the current provenance summarization methods fail to ensure high relevance between the provided explanations and user questions. To solve the above issues, we propose a new method that uses the relevance between query results and their context to improve provenance summarization. Our method assigns important labels to all generated summaries, ensuring users can get effective explanations of the question through the provenance summarization method, regardless of their prior knowledge. Additionally, we introduce an attribute filtering method that, through multiple stages, filters different types of attributes to address issues of low algorithm efficiency and poor explanation quality caused by certain attributes. Finally, comprehensive efficiency and quality experiments were conducted on three large-scale real-world datasets. The results demonstrate that our approach not only significantly enhances the efficiency of provenance summarization algorithms but also effectively improves summary quality across multiple metrics.

PVLDB Reference Format:
Hongtao Song, Mingxuan Liu, Qilong Han, and Zhiwen Yu. Context-Augmented Impact Summary with Attribute Filtering for Provenance Summarization. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

*Qilong Han is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/1789686787/CAISG-MAF.

1 INTRODUCION

Today's world is a data-driven era of big data, where people mine huge amounts of complex data for potential and valuable information to guide various tasks. Provenance technology [9, 22] can be used to describe the origins of process data and all the information that generated it, making it the only technique currently ensuring data authenticity. Provenance data is crucial for interpreting process data, to the extent that scholars are beginning to explain relational database query results by mining or aggregating provenance data.

The existing methods of aggregate query provenance analysis, which explain query results by summarizing provenance information [4, 26], have been acknowledged to yield valuable and sometimes surprising insights. However, we contend that this is insufficient. For instance, if a user seeks to explore the characteristics of the hundreds of thousands of comedies released to date, existing provenance analysis systems may offer two features: a length of 90 to 120 minutes and English language. Nonetheless, this may not align with user expectations, as these attributes are common across most movie genres. Users are more inclined to discover unique comedic features compared to other film types. In other words, the relevance of attributes such as a length of 90 to 120 minutes and English language to the user question is relatively low.

Now, let us provide a brief introduction to some fundamental concepts influencing the entirety of this paper. In this study, we will utilize summaries comprising multiple feature rules to explicate specific user questions. Feature rules are tuples containing attribute values and placeholders (referred to as "rules" for brevity), to which we assign the attributes of 'public' or 'private', denoting them as public features and private features, respectively. In essence, public features denote shared information among various sub-entities of an entity. For instance, the duration of a film being 90-120 minutes is a common trait, scarcely influenced by the genre of the movie. Private features, on the other hand, signify unique attributes of a sub-entity in comparison to others. For instance, during a certain period, the gender of directors of war films was predominantly male. While public features offer guidance, they fall short in accurately delineating distinctions among sub-entities of an entity, whereas private features are more conducive to elucidating specific user questions. Owing to the robust correlation evident among

Table 1: Movies

No	Titleid	Year	Genre	GenreId	Rating	Directorid	Gender	Runtime
t_1	tt001012	2008	action	1	8.5	nm000510	male	long
t_2	tt001013	2007	drama	3	6	nm000511	female	long
t_3	tt001014	2008	action	1	7.5	nm000512	female	normal
t_4	tt001015	2008	drama	3	6	nm000513	female	short
t_5	tt001016	2006	drama	3	4.5	nm000514	female	short
t_6	tt001017	2008	action	1	8	nm000515	female	normal
t_7	tt001018	2008	action	1	8.5	nm000516	male	normal
t_8	tt001019	2007	comedy	2	8	nm000517	male	short
t_9	tt001020	2007	comedy	2	8.5	nm000518	male	short
t_{10}	tt001021	2008	action	1	7.5	nm000519	male	normal
t_{11}	tt001022	2008	action	1	8	nm000520	female	short
t_{12}	tt001023	2008	action	1	7	nm000521	female	long
t_{13}	tt001024	2008	horror	4	8	nm000522	male	normal
t_{14}	tt001025	2006	music	5	8	nm000523	male	normal

Table 2: Results of Q_1

No	Gender	AvgRating
a_1	male	8.14
a_2	female	6.71

Table 3: Impact summary

No	Year	Genre	GenreId	Gender	Runtime	Score
r_1	2008	action	1	male	⋆	0.92
r_2	2007	comedy	2	male	⋆	0.36
r_3	2006	music	5	male	normal	0.15

Table 4: Impact summary of context

No	Year	Genre	GenreId	Gender	Runtime	Score
r_4	2008	action	1	female	⋆	2.36
r_5	⋆	drama	3	female	⋆	1.83

Table 5: context-augmented Impact Summary

No	Type	Year	Genre	Gender	Runtime	Score	GenreId
r_6	public	2008	action	male	⋆	1.15	1
r_7	private	2007	comedy	male	⋆	0.45	2
r_8	context	⋆	drama	female	⋆	2.44	3

real-world data, we have discerned that the knowledge distinguishing whether feature rules are public or private resides within the context of user questions.Therefore, In this paper, we aim to address this issue by mining contextual information embedded within user questions. Fundamentally, we propose annotating explanations to aid users in distinguishing their relevance to the questions. The formal exposition of the concept of context will be provided in subsequent sections. Furthermore, existing methods of aggregate query provenance analysis solely employ filters for attributes with nearly unique values, thus neglecting numerous other types of low-value attributes. Consequently, deriving valuable summaries effectively becomes challenging. However, we have discovered that employing attribute classification and multi-stage filtering can effectively eliminate low-value attributes. Subsequently, we will provide a concise overview of existing provenance summarization methods and outline our primary research efforts.

Example 1.1. Consider a relational database user executing a standard *SQL* aggregate query, denoted as Q_1, on Table 1, a sample dataset from *IMDB*[1] containing 14 tuples. The primary objective of this query is to retrieve the average ratings of movies directed by individuals of varying genders. Subsequently, Table 2 illustrates the outcomes of Q_1, indicating an evident trend where movies directed by male filmmakers yield higher average ratings compared to their female counterparts. To explain this phenomenon, one may designate tuple a_1 as the user question. Employing data provenance techniques, we can trace the query result a_1 back to its origins, thus deriving a tuple set $\{t_1, t_7, t_8, t_9, t_{10}, t_{13}, t_{14}\}$ from Table 1. Consequently, users stand to benefit from these tuples. Nevertheless, the complete *IMDB* dataset comprises millions of records, necessitating a summarized representation of the provenance tuples. Users would derive greater utility from such condensed summaries rather than exhaustive provenance data.

Q_1: **SELECT** Gender, **AVG**(Rating) **As** AvgRating
FROM Movies **GROUP BY** Gender

We aim to provide an explanation for the user query a_1, which involves elucidating why the average rating of movies directed by male directors is 8.14. To describe this task, we first use an impact-based summarization approach to generate the top-3 impact summaries [4]. As illustrated in Table 3, the top-3 impact summary

[1]https://www.imdb.com

consists of three feature rules denoted as $\{r_1, r_2, r_3\}$, where each feature rule comprises specific attribute values and a wildcard $\star$. The wildcard $\star$ signifies a match for any attribute value; for instance, r_1 can match all tuples in Table 1 with $Year = 2008$, $Genre = Action$, $GenreId = 1$, and $Gender = male$, i.e., $\{t_1, t_7, t_{10}\}$. The score within the impact summary is utilized for rule evaluation. Therefore, based on this impact summary, it appears that a conclusion can be drawn: the rule $(2008, action, 1, male, \star)$ constitutes a primary factor contributing to the high ratings of movies directed by male directors, indicating that male directors produced a greater number of highly rated action films in 2008. However, upon further exploration of Table 1, it appears that the facts may not align as initially perceived.

Table 4 illustrates the impact summary of a_2, with the unexpected observation that the feature rule $(2008, action, 1, female, \star)$ achieves the highest score. This indicates that female directors also directed a substantial number of highly rated action films in 2008. Combined with the findings from Example 1.1, it appears that both male and female directors directed numerous highly rated action films in 2008. Therefore, the feature rule $(2008, action, 1, male, \star)$ cannot precisely explain why the average ratings of movies directed by male directors were higher than those directed by female directors in 2008. Perhaps the rules r_2 or r_3 in Table 3 are crucial to understanding the issue. However, for users without prior knowledge, it is nearly impossible to discern this fact. Below is a brief introduction to the solution we use in this paper.

The impact summary of a_2 in Table 4 serves as contextual information to augment the impact summary of a_1. Table 5 presents the context-augmented impact summary of a_1, where it can be observed that we introduce a new attribute named $Type$ compared to the impact summary, which is utilized to describe the type of rules, with possible values being $\{public, private, context\}$. Here, $public$ indicates that the feature rule is a common attribute across the entire dataset and is unrelated to the user question, such as the previously mentioned r_1 and r_4. We refer to such rules as public features and set their $Type$ attribute to $public$. $Private$ signifies that the rule is a private feature. For instance, the feature rule r_2 $(2007, comedy, 2, male, \star)$ appears to be an exclusive rule for male directors, which is more relevant to the user question. For rules providing insufficient information, we propose a substitution approach, replacing low-value rules with high-scoring private rules from the context. For this type of rules, we set their $Type$ attribute to $context$. Subsequently, we will define the complete rule set used to generate the context-augmented impact summary.

In addition, the attribute *GenreId* is moved to the end in Table 5, and we can observe from Table 1 that all *Genre* and *GenreId* attribute values have a one-to-one correspondence in this sample, e.g., $action - 1$, $comedy - 2$, $drama - 3$. This means that the attribute *GenreId* can be used as a derived attribute of *Genre* on this sample, then we do not need the *GenreId* in the calculation. We also filter out near-unique attributes such as $Titleid$, $Directorid$, which can cause the summarization task to not work properly, and we propose a multi-stage attribute filtering method to remove multiple low-value attributes including that mentioned above.

The main contributions of our work are as follows:

- As far as we know, we are the first to put contextual information into provenance summarization and propose context-augmented impact summary, as well as the formal definition of the context-augmented impact summarization problem.
- We define three types of low-value attributes that need to be filtered in the provenance summarization task.
- We propose the efficient algorithms to filter attributes, as well as to generate context-augmented impact summaries.
- We conduct thorough efficiency experiments as well as quality experiments on three datasets to prove that our solution is feasible and useful.

2 RELATED WORK

2.1 Data Provenance

In recent years, data provenance has been widely studied in the database field, and relational database queries have various forms of provenance [10, 28], including why-provenance [9, 14, 16, 17, 43] describes source data that influence on the existence of the data. Why-not provenance [8, 12, 30] is used to explain why the query results are missing. The information about how to obtain output tuples based on a query is called how-provenance [24]. Where-provenance [14] describes where a set of data originates from the source database. Lineage lists the tuples that cause a tuple to appear in a query answer [15]. Considering the substantial cost associated with managing lineage information in practical DBMS, various provenance management frameworks have been proposed in literatures [5, 13, 29, 35] to store and retrieve pertinent provenance information. There are several frameworks that support why-provenance for the relational database query, such as *Perm* [23], *GProM* [6] and *ProvSQL* [40]. In this paper, *Perm* is used to generate the provenance of aggregate query results, which can mark the results and obtain the provenance of query results through a query rewriting approach.

2.2 Database Aggregate Query Explanation

Explaining aggregate query results is a challenging research topic, and various forms of explanation have been proposed in past studies. The origins of research in this field can be traced back to the emergence of OLAP [39] systems, and in recent years, studies pertaining to OLAP systems have gained significant traction. For the explanation of outliers, current research has mainly focused on generating predicate patterns [32, 33, 37, 38, 44]. *Scorpion* [44] can find explanations based on the user-specified outliers, and *Cape* [33, 34] explains the outliers in aggregate query results according to the concept of counterbalance, and *Cajade* [32] generates the join graphs based on database schema graphs, which in turn mines for valuable explanations. Early research on data provenance predominantly leaned towards employing a summarization technique to represent approximate provenance [3, 31, 36]. The summarization of query provenance prefers to summarize high-value features in the data to guide users rather than to focus on explaining outliers [4, 26, 41]. *SmartDrill − Down* [25] is used to explore and summarize interesting tuples and supports drill-down operations. *QAGView* [42] directly summarizes the top-L tuples in the table. Inspired by the concept of counterbalance in [33], we first put the context into the study on provenance summarization.

2.3 Summarization Rules

The summarization rule is used to describe how to summarize the set of tuples and obtain a summary. The concept of summarization was first introduced in [26] and it was proved that the summarization problem is np-hard. [41] uses three parameters to constrain the rules in the summary and obtain a high score summary. [4] used sensitivity analysis [27] to calculate the impact instead of coverage to calculate the score, which has better performance under the aggregate functions SUM and AVG. In addition, result diversification, summarization and exploration have been widely studied in the literature and applied to the database field [1, 2, 11, 18–21]. In this paper we propose a new summarization rule for generating high quality context-augmented impact summaries.

3 PRELIMINARIES

In this section we give a formal description of our presented solution. All the symbols that we use are summarized in Table 6.

3.1 Basics

In order to better describe our solution, we first review some basic concepts and definitions.

3.1.1 Aggregate Queries. In this paper, we focus primarily on aggregate queries as our main research topic, given that the majority of queries in relational databases can be categorized as aggregate queries. We define a general standard SQL aggregate query Q as follows. Here, $Table$ is a base relation in database D, or it could be the result of a nested subquery. This approach ensures that our method can be extended to aggregate queries involving multiple relational tables. $Attr = \{A_1, ..., A_m\}$ is the set of all attributes of $Table$. Agg represents standard aggregate functions, which can include basic SQL aggregate functions such as SUM, AVG, COUNT, MIN, and MAX, as well as user-defined aggregate functions available within SQL statements. The result of Q is $\{a_1, ..., a_n\}$.

Q: **SELECT** A_1, ..., A_m, **AGG**(A^*) **As** AggValue
FROM Table <relational table or output of a nested query>
WHERE Conditions
GROUP BY A_1, ..., A_m

To simplify future formulas, it is necessary to introduce the concept of $Q^a(Table)$. $Q^a(Table)$ denotes the aggregate value corresponding to result a after executing Q on $Table$, where $a \in \{a_1, ..., a_n\}$. Alternatively, it can be understood as the result of executing a specific aggregate function, as demonstrated by Q^a. For example, for the query Q_1 in Example 1.1, $Q_1^{male}(Movies)$ represents the aggregated value within the *male* tuple after executing Q_1, specifically $Q_1^{male}(Movies) = a_1 = 8.14$.

Q^a: **SELECT AGG**(A^*)
FROM Table <relational table or output of a nested query>
WHERE Conditions AND $A_1 = a\,[A_1]$ AND...$A_m = a\,[A_m]$

3.1.2 Data Provenance. Data provenance facilitates tracking the contributing original tuples to query results. Here, a simple why-provenance model suffices for our needs. In this paper, we utilize the *Perm* [23] framework for source retrieval due to its superior performance in retrieving the origins of aggregate queries and nested subqueries.In the example, we will illustrate using simple aggregate queries, hence the source tuples can be regarded as the tuples that influence the results of the aggregate queries. $PT(a^*)$ denotes the provenance table of a^*, which is the tuple of questions selected by the user, and $\{t_1, ...t_k\}$ is the set of tuples in the provenance table. For example, for $a^* = male$ in Example 1.1, $PT(male) = \{t_1, t_7, t_8, t_9, t_{10}, t_{13}, t_{14}\}$.

3.1.3 Summarization Rules. The result of provenance summarization is a summary, which should contain k feature rules in the top-k question, and the rule is the basic unit of the summary. A rule is a tuple of length $|Attr|$ and each value in the tuple is an attribute value or a wildcard $\star$, where $\star$ matches all the attribute values. We measure the value of a summary by computing its score, which depends on the impact and weight of the rules it contains, as shown in Equation 1, where S represents a summary. However, as mentioned earlier, high-scoring rules merely represent their aggregation of numerous high-impact source tuples, which does not necessarily prove their ability to effectively explain user questions.

$$Score(S) = \sum_{rule \in S} Impact(rule) \times Weight(rule) \tag{1}$$

3.1.4 Weighting Function. $Weight(rule)$ is the weight of the rule, which is used to measure the amount of information expressed by the rule. In [26], detailed methods for selecting the weight function of the rule are compared. Here, we introduce three of the most common weighting functions. As shown in Equation 2, the size weighting function determines the weight value by calculating the number of non-$\star$ values in the rule. For example, in Table 3, $Weight(r_1) = 4$, $Weight(r_2) = 4$, $Weight(r_3) = 5$. The advantage of this weighting function is that it treats all attributes equally without adding subjective factors, and it does not require additional pre-evaluation of attributes. However, in some cases, this may seem unreasonable because it may lead to the neglect of certain important attributes that should have appeared in the rule. In the absence of any prior knowledge, the size weighting function remains the best choice.

$$Weight(Rule) = \sum_{A_i \in Attr} \mathbb{I}(A_i[rule] \neq \star) \tag{2}$$

The remaining two weighting functions, as depicted in the equation, are both weighted based on the quantity of attribute values corresponding to attributes. Here, $dom(A_i, T)$ represents the domain of attribute A_i in the relation table T. In fact, they are not commonly used in research, as inevitably in data, there exist some attributes with nearly unique original attribute values (such as id, name, etc.). In which case using these two weighting functions would result in inferior summaries. However, if users possess prior knowledge, these two weighting functions are evidently superior to the size weighting function.

$$Weight(Rule) = \sum_{A_i \in Attr} \log_2(|dom(A_i, T)|) \tag{3}$$

$$Weight(Rule) = \sum_{A_i \in Attr} \sqrt{|dom(A_i, T)|} \tag{4}$$

Table 6: symbols and notations

Symbol	Descirption
Q	A standard aggregate query.
$Q^a(Table)$	Aggregate value of a in Q
$D, Table$	A database and the original table in the database
$Attr = \{A_1, A_2.....A_n\}$	A set of attributes
$\{a_1, a_2.....a_n\}$	The result of Q
a^*	A user question tuple
$PT(a^*)$	Provenance tables for a^*
$\{t_1, ...t_k\}$	The set of tuples in $PT()$
$t_i[A]$	The attribute value of A in t_i
$dom(A)$, $dom(A, T)$	The domain of A and the domain of A on table T
r	A feature rule
S	A provenance summary
K	The maximum size of a summary
$\star$	Placeholder
lbw , ubw	The upper and lower bounds for the weight of the rules
$\lambda_{dispersive}$	Threshold for filtering the dispersive attributes
λ_{object}	Threshold for filtering the objective attributes
$\lambda_{dependent}$	Threshold for filtering the dependent attributes

3.1.5 Coverage And Impact. Before introducing Impact, we first present the concept of Coverage. The concept of Coverage is obvious, referring to the set of original tuples covered by rules, used to describe the extent to which rules summarize the original data. For a considerable period in past research, Coverage and weighting functions were directly employed to compute the Score of rules, until the concept of Impact emerged. The computation method of Coverage is as depicted in the equation 5, where we introduce the concept of child-rule for definition. For two rules r_a and r_b, if r_a can be obtained by replacing specific attribute values with $\star$, then r_a is termed a child-rule of r_b.

$$Coverage\ (rule) = \left\{t_i | t_i \in PT(a^*),\ t_i \text{ is the child–rule of } rule\right\} \tag{5}$$

Example 3.1. For $r_1(2008, action, 1, male, \star)$ in Example 1.1, we can get $t_1(2008, action, 1, male, long)$ by replacing $\star$ with $long$, so t_1 is the child-rule of r_1. similarly, t_7, t_{10} are also the child-rule of r_1. i.e. $Coverage(r_1) = \{t_1, t_7, t_{10}\}$.

The $Impact\ (rule)$ is the degree of that the rule contributes to the aggregate value in the results, which is determined by the impact of t_i in $Coverage\ (rule)$. The impact of t_i can be calculated using *sensitivity analysis* [27]. As is shown in Equation 6, it is determined by the difference of $Q^a\ (Table)$ between t_i in and out of table. It should be noted that there are multiple rules in the summary, and tuples that have been covered by previous rules should not be calculated repeatedly in the later ones. Here, it is represented by the $Coverage'\ (rule)$.

For queries that aggregate numerical attributes using functions such as SUM and AVG, it seems that Coverage cannot accurately depict the contribution of tuples to the aggregated value. In other words, the contribution of multiple tuples to the aggregated value may not necessarily exceed that of a specific tuple. Here, we employ a straightforward sensitivity analysis method to illustrate this scenario, as it effectively evaluates the influence of input values on outcomes. For instance, considering tuple t_{10} in Table1, if we remove it from the table, the aggregated value of tuple a_1 in Table2 will become $\frac{(57-7.5)}{6} = 8.25$, indicating that the impact of t_{10} on the aggregated value is $|8.14 - 8.25| = 0.11$. Similarly, if we consider the removal of tuples t_8, t_{13}, and t_{14} from Table1, the aggregated value of a1 will become $\frac{(57-8-8-8)}{4} = 8.25$, with an impact of $|8.14 - 8.25| = 0.11$. It is evident that for the aggregated value of a1, the contribution of $t10$ is equivalent to the cumulative contribution of t_8, t_{13}, and t_{14}. It is noteworthy that for the aggregation function $COUNT$, since the contribution of tuples to the aggregated value is fixed at 1, the above situation does not arise.

As demonstrated in Equation 6, the Impact is determined by the disparity of $Q^a\ (Table)$ for t_i being present and absent in the table. It is noteworthy that multiple rules exist in the summary, and tuples covered by prior rules should not be redundantly calculated in subsequent rules. For ease of distinction, we employ $Coverage'\ (rule)$ to represent this concept.

$$Impact\ (rule) = \sum_{i=1}^{|Coverage'(rule)|} |Q^{a^*}(Table) - Q^{a^*}(Table \setminus \{t_i\})| \tag{6}$$

Example 3.2. For feature rule r_1 in Example 1.1, according to Equation 6, $Impact(r_1) = |Q_1^{male}(Movies) - Q_1^{male}(Movies \setminus \{t_1\})| + |Q_1^{male}(Movies) - Q_1^{male}(Movies \setminus \{t_7\})| + |Q_1^{male}(Movies) -$

Q_1^{male} $(Movies \setminus \{t_{10}\})|$, and $Q_1^{male}(Movies)$ is the average rating of movies directed by male directors in the Moives table i.e. Q_1^{male} $(Movies) = a_1\,[AvgRating] = 8.14$, $Q_1^{male}(Movies \setminus \{t_1\})$, Q_1^{male} $(Movies \setminus \{t_7\})$, $Q_1^{male}(Movies \setminus \{t_{10}\})$ are the average after removing t_1, t_7, t_{10} respectively which are calculated as 8.08, 8.08, 8.25. $Weight(r_1)$ is the number of non-$\star$ values in the r_1 that are 4, so according to Equation 1 $Score(r_1) = Impact(r_1) \times Weight(r_1) = 0.92$.

3.2 Context-Augmented

In this section, we commence the formal introduction of context enhancement methods, which encompass the formal definition of context, followed by the distinction between public features and private features based on contextual information.

3.2.1 Context. In the field of computer science, the concept of context is broad, necessitating explicit definition based on specific environments and processes. The context mentioned in this paper refers to the set of remaining tuples in the aggregated query result, excluding the user question tuple a^*, denoted as $Q(Table) \setminus \{a^*\}$. A wealth of previous research has demonstrated the significant importance of context in explaining problems. Influenced by the notion of counterbalance as depicted in [33], we posit that context can serve as a crucial complement to the interpretation of user questions. To facilitate subsequent discussions on the application of context, we introduce the concept of context summary. For the user question tuple a^*, we denote $S'(a^*)$ as the context summary, representing the entirety of context as a summary generated from the user question. The computation process of context summary bears similarity to that of general summaries, and its result also yields top-k scoring rules.

3.2.2 Public Feature And Private Feature. If we extract features shared between the context and the user question, this may suggest that such features cannot serve as explanations for distinguishing between the context and the user question, indicating a poor relevance to the user question. However, this does not imply that public features lack value, for users without any prior knowledge, public features remain important ways for understanding the data. Therefore, in this paper, we solely mark feature rules, providing users with an essential means to discern features. Nonetheless, distinguishing between public and private features is challenging to quantify. In this paper, we adopt a summary-based comparison method to define public features, as shown in Definition3.3.

Definition 3.3. [Public Feature] Given a relational table or nested query sub-table T, a standard aggregation query Q , a user query a^*, and a positive integer K , generate top-k summaries S and $S'(a^*)$ using a^* and $Q(T) \setminus \{a^*\}$ respectively. If there exists $r_i = r_i'$ or r_i' is a child-rule of r_i , and $r_i \in S$, $r_i' \in S'(a^*)$, then we consider r_i to be a public feature.

3.2.3 Singleton Rules. In the preceding discussion, we introduced rules consisting of attribute values and $\star$. When a rule lacks $\star$, it degenerates into a tuple, which we term a singleton rule. If singleton rules are presented to users, at this point, provenance summarization degrades into basic data provenance methods. Due to the minimal information content provided by singleton rules, they scarcely contribute to problem explanation. In this paper, we aim to propose an approach employing high-value contextual rules to substitute those offering minimal information. We place the emphasis of problem explanation on distinguishing the differences between user question and context, believing that the rules included within contextual rules can similarly aid users in problem explanation. In essence, we will set a constant k', determining the quantity of contextual rules to replace, with specific replacement rules provided in section 4.

3.3 Provenance Summarization Problem

Before proposing the algorithm, it is necessary to formally define the problem. In this subsection, we first review the impact summarization problem discussed in [4], and then provide a formal definition of the context-augmented impact summarization problem.

3.3.1 The Impact Summarization Problem. High-score impact summary can be generated by calculating impact and weight, and the impact summarization problem is defined as follows.

Definition 3.4. [The Impact Summarization Problem] Given one of the result tuples of a aggregate query a^*, a provenance table $PT(a^*)$ and a constant K, the impact summarization problem is to find a summary S of size K such that $Score(S)$ is maximized.

3.3.2 The Context-augmented Impact Summarization Problem. We propose the context-augmented impact summary based on the context of the query result, and to indicate the type of the rule, an additional attribute column $type$ to each rule is necessary, where $type \in \{public, private, context\}$. The $public$ means that the rule is a public feature on the whole original table. The $private$ means that the rule is a private feature of $PT(a^*)$. The $context$ represents that the rule is a replacement of the original rule with contextual information.

Definition 3.5. [The Context-Augmented Impact Summarization Problem] Given S_1 is the impact summary of the user question a^*, S_2 is the impact summary of $Q(Table) \setminus \{a^*\}$, and a constant K'. The context-augmented impact summarization problem is to label all rules in S_1 and S_2 as public or private, and find no more than K' r_j marked as private to replace r_i to maximize $Score(S_1)$, where $r_i \in S_1$ and $r_j \in S_2$. The S_1 labeled and replaced is referred to as the context-augmented impact summary.

3.4 Attribute Filtering

The problem of summary generation has been proven to be NP-Hard[25], and current algorithms for handling the summarization problem are heuristic algorithms. Therefore, the selection of attributes has a significant impact on the efficiency of the algorithms, and in some cases, it may degrade the quality of the summary. We aim to perform filtering operations on the original attributes of the data before inputting the data into the summarization algorithm. In this chapter, we will focus on defining which attributes should be filtered out before inputting the algorithm.

3.4.1 Dispersive Attributes. Those nearly unique attributes in the original data hardly contribute to the summary. For example, each

value in $Titleid$ and $Directorid$ in Table 1 is unique, and if appearing in rule, the attribute will be $\star$ values forever. We should filter attributes with excessively decentralized domains, as they do not contribute to the summary and significantly reduce the efficiency of the summarization algorithm. The definition of dispersive attributes is as shown in Definition3.6., where $dom(A, T)$ represents the domain of attribute A in table T.

Definition 3.6. [Dispersive Attribute] Given a attribute A, table $PT(a^*)$ and a fixed threshold $\lambda_{dispersive}$, we call A a dispersive attributes if $|dom(A, PT(a^*))| > \lambda_{dispersive}$.

3.4.2 Objective Attributes. If a certain attribute value occurs with high frequency, it is highly probable that all top-k rules in the summary will include this attribute value. For instance, we observed an attribute named "isAdult" in the IMDB dataset, with possible values $\{0, 1\}$; however, about 96.89% of tuples in the complete dataset have a value of 0 for "isAdult". We attempted to execute summarization algorithms on the dataset, and the results revealed that all generated rules had an "isAdult" value of 0. This indicates an extremely poor relevance of this attribute value to any user question. Therefore, we intend to regard such attribute values as objective common knowledge and exclude them from the summarization algorithm. The formal definition of objective attributes is as shown in Definition3.7.

Definition 3.7. [Objective Attribute] For a classification attribute A and a fixed threshold λ_{object}, if $\exists t_i\,[A] \in dom(A, PT(a^*))$ satisfies $\frac{|Coverage(\star,....t_i[A]...,\star)|}{|Coverage(\star,....,\star)|} > \lambda_{object}$, we call attribute A an objective attribute.

3.4.3 Dependent Attributes. In the original attributes of the dataset, there are often pairs of attributes that consistently occur together, such as birth year and age, city name and street name, director name and director gender, etc. We observe a unidirectional dependency within such attribute pairs, thus in the summarization algorithm, it is sufficient to input only those dependent attributes. This operation not only does not affect the final summarization results but also enhances the efficiency of the algorithm. The definition of dependent attributes is as shown in Definition3.8.

Definition 3.8. [Dependent Attribute] For any two attributes A_i, $A_j (i \neq j)$, $byies(T, A_i, A_j)$ is a Bayesian network of A_i pointing to A_j built using table T. Given a fixed threshold $\lambda_{dependent}$ and $byies(PT(a^*), A_i, A_j)$, if $\forall t\left[A_j\right] \in dom(A_j, PT(a^*))$, $\exists P(t\left[A_j\right] \mid t\,[A_i]) > \lambda_{dependent}$ ($P(|)$ represents the conditional probability), we call A_j a dependent attribute of A_i.

4 ALGORITHMS

In order to filter the attributes and generate context-augmented impact summaries, we design the multi-stage attribute filtering algorithm (*MAF*) and the context-augmented impact summary generation algorithm (*CAISG*), respectively. Besides, the cost of the algorithms will be discussed.

4.1 IPS Algorithm

In order to facilitate the understanding of our proposed algorithms, we first review the implementation of the *IPS* [4] algorithm, which is used to find the sub-optimal solution to the impact summarization problem and can optimize the efficiency of the algorithm execution using pruning technique. Algorithm 1 describes the basic calculation of the *IPS* algorithm. In order to select the rules with top-k scores, the algorithm needs to go through K selections and each time select the rule with the highest current score(Line 1-2). In the process of selecting the highest scoring rule, we define two rule sets S_n and S_0 in the first, S_n contains the rules with ownership weight $j-1$ and S_0 contains the rules with ownership weight j. Using S_n to generate its child-rule in each iteration round, rules with weight-j are generated and added to S_0 until the specified upper bound of weight ubw is reached(Line 5-10). In the *IPS* algorithm, we establish a lower bound of lbw and an upper bound of ubw for the weight of the rules. This decision is driven by our objective to strike a balance between providing adequate information and avoiding excessive length in the generated rules.

In addition, the pruning technique can be used to improve the efficiency of algorithm execution. If $Impact(r) \times ubw$ is smaller than the rule with the highest current score for any rule r, then r and all its $child-rule$ are not possible candidates(Line 11-12). For eligible rules, the rule s with the largest current score will be selected and added to S. Then all tuples covered by s in table T will be removed to ensure that s is different for each round of selection(Line 15-20).

Algorithm 1: The *IPS* Algorithm.

Input: Table T, query Q, user's question a^*, attribute set $Attr$, K.
Output: A set of rules S.

```
1  S = φ;
2  for i from 1 to K do
3      S_n = all rules with weight 1;
4      maxscore = the maximum Score(s_n) , s_n ∈ S_n;
5      for j from |lbw| to |ubw| do
6          S_0 = φ ;
7          foreach s_n ∈ S_n do
8              R = all weight-j child-rule of s_n;
9              foreach r ∈ R do
10                 Impact(r) =the impact of rule r;
11                 if Impact(r) × |ubw| < maxscore then
12                     continue;
13                 else
14                     S_0 = S_0 ∪ {r};
15                     maxscore = MAX(maxscore, Score(r));
16                     if Score(s) < Score(r) then
17                         s = r
18             S_n = S_0;
19     S = S ∪ {s};
20     T = T \ {Coverage(s)};
21 return S
```

4.2 Context-Augmented Impact Summary Generation Algorithm

Next, we explore the generation of context-augmented impact summaries. In this paper, our focus lies in distinguishing and replacing

existing summaries. Therefore, we abstain from discussing optimizations pertaining to heuristic summary generation algorithms. In other words, our aim is to ensure that our algorithm can accommodate existing summaries, making it adaptable to diverse pre-existing summarization schemes.As depicted in Algorithm 2, we initially acquire the provenance tables T and T' of the user question a^* and its context, obtained using the perm[23] framework (Lines 1-2). Subsequently, we employ the *IPS* algorithm to generate their impact summaries, denoted as S and S', respectively (Lines 3-4). Then, we proceed to ascertain whether all rules in S and S' are public or private features, as described in Definition 3.3 (Lines 5-16). Specifically, we iterate through each *rule* in S. If there exists a $rule'$ in S' identical to *rule*, both *rule* and $rule'$ are considered public features. If $rule'$ is a child-rule of *rule*, then only *rule* is determined to be a public feature. Any remaining undetermined $rule'$ is assumed to be a private rule (lines 14-16).

In lines 17 to 23, we will replace the lowest-scoring *rule* to achieve the objective of maximizing the score as defined in3.5. We will employ a greedy strategy, where we replace the lowest-scoring *rule* with the highest-scoring private feature $rule'$, and this replacement process will be carried out for a maximum of a constant k' times.

Algorithm 2: The *CAISG* Algorithm.

Input: Database D, query Q, user's question a^*, attribute set $Attr$, K, K'.
Output: A context-augmented impact summary with K rules.

1 $T = PT(a^*)$;
2 $T' = PT(Q(T) \setminus \{a^*\})$;
3 $S = IPS(T, Q, a^* \in Q(T), K, Attr)$;
4 $S' = IPS(T, Q, Q(T) \setminus \{a^*\}, K, Attr)$;
5 **foreach** *rule in S* **do**
6 **foreach** *rule‘ in S′* **do**
7 **if** $rule' = rule$ **then**
8 $rule(type) = public$;
9 $rule'(type) = public$;
10 **else if** $rule'$ *is a child-rule of rule* **then**
11 $rule(type) = public$;
12 **else**
13 $rule(type) = private$;
14 **foreach** *rule‘ in S′* **do**
15 **if** $rule(type)! = public$ **then**
16 $rule(type) = private$;
17 **while** *i from 1 to K′* **do**
18 $MinRule$ = lowest-score rule in S;
19 $MaxRule$ = highest-score private rule in S';
20 **if** $Score(MinRule) < Score(MaxRule)$ **then**
21 $MaxRule(type) = context$;
22 Replace $Minrule$ in S with $MaxRule$;
23 delete $MaxRule$ from S';
24 **return** S

4.3 Multi-Stage Attribute Filtering Algorithm

The *MAF* algorithm contains three stages, each of them filters different types of attribute. The *MAF* algorithm pseudocode is shown in Algorithm 3, and to improve the readability, we omit some simple details.

The three stages of attribute filtering are independent of each other. Each stage filters the low-value attributes as defined in Section 3.4, and we keep updating the value of $Attr'$ until all low-value attributes have been removed. The first two stages calculate only the attribute domain, including recording the domain length and counting the attribute values. The third stage requires comparing each pair of attributes and determining whether there is an attribute dependency by calculating the conditional probability.

Algorithm 3: The *MAF* Algorithm.

Input: Orginal attribute set $Attr$, table T, $\lambda_{dispersive}$, λ_{object}, $\lambda_{dependent}$.
Output: Filtered attribute set $Attr'$.

1 $Attr' = Attr \setminus DispersiveAttr(Attr, T, \lambda_{dispersive})$; // Call $DispersiveAttr()$ to iterate through the set $Attr$ and find all dispersive attributs as defined in definition 3.6.
2 $Attr' = Attr' \setminus ObjectiveAttr(Attr', T, \lambda_{object})$; // Call $ObjectiveAttr()$ to iterate through the set $Attr'$ and find all objective attributs as defined in definition 3.7.
3 $Attr' = Attr' \setminus DependentAttr(Attr', T, \lambda_{dependent})$; // Call $DependentAttr()$ to iterate through all attribute pairs in the $Attr'$ set and find all dependent attributs as defined in definition 3.8.
4 **return** $Attr'$

4.4 Algorithm Analysis

The time cost of *MAF* algorithm mainly depends on the three stages of attribute filtering. The first two stages mainly contain the computation of each attribute domain, which costs $O(n)$ time. Here, n represents the size of the table. The total cost of the first two stages does not exceed $O(m \times n)$, where m represents the number of original attributes. In the third stage, we need to build the Bayesian network for each pair of attributes, and it costs $O(n)$ time for each pair. The number of attribute pairs do not exceed $\frac{m(m+1)}{2}$, so the cost of the third stage does not exceed $O(m^2 \times n)$. Then, the total cost of *MAF* algorithm does not exceed $O(m^2 \times n)$. The cost of the *CAISG* algorithm mainly comes from the *IPS* algorithm, because the cost of the other parts in *CAISG* is almost ignorable compared to the *IPS* algorithm. Therefore, the total cost of the *CAISG* algorithm does not exceed $O(K \times n^2 \times m)$ [4].

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setup

We built a complete system for evaluating our algorithm in Python 3.6, which runs on a machine with a 2.6GHz Inter CPU and 10GB RAM. The PostgreSQL is chosen as the database and the *Perm* [23] is used to generate the provenance of aggregated query results.

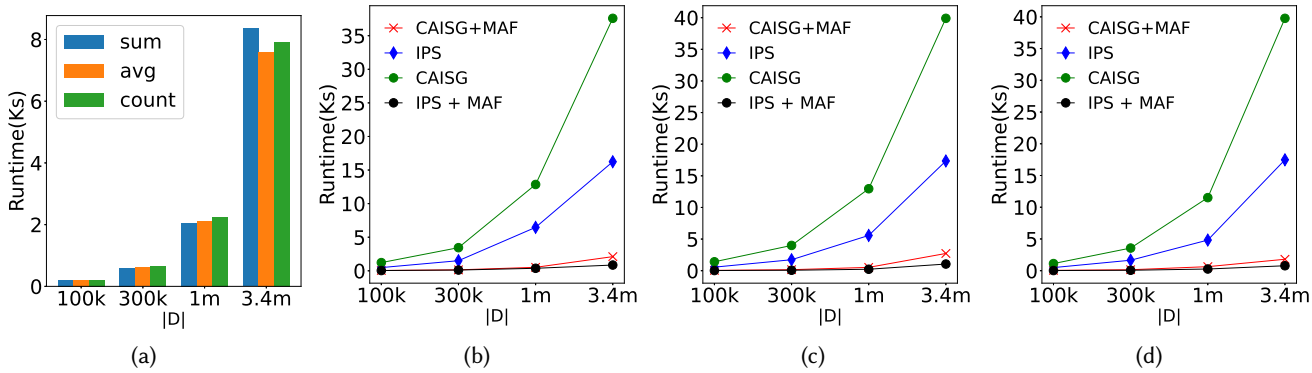


Figure 1: Execution of four different algorithms containing IPS , IPS+MAF, CAISG, CAISG+MAF algorithms on TPC-H datasets of different sizes, (a) algorithm runtime versus database size; (b) aggregate function = AVG; (c) aggregate function = SUM; (d) aggregate function = COUNT. The running time of different aggregate functions is similar.

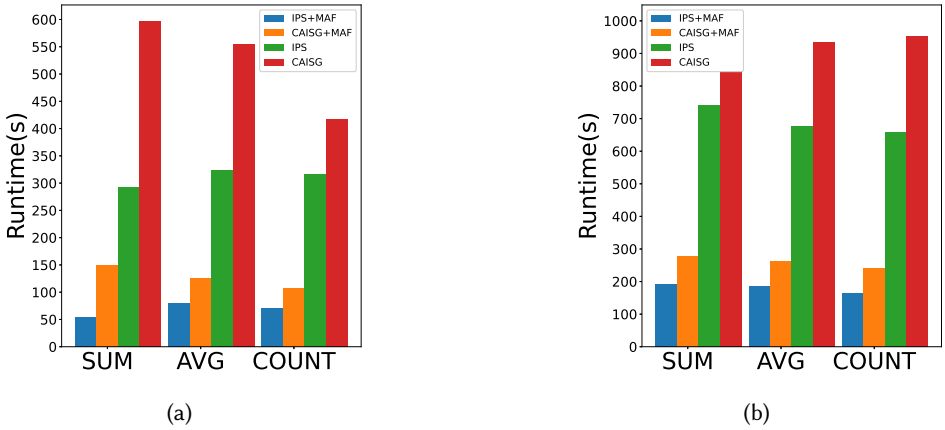


Figure 2: Runtime of four different algorithms on IMDB and Census Income. (a) IMDB; (b) Census Income. MAF algorithm can effectively reduce the runtime of the algorithm.

Table 7: Datesets

	Census Income	IMDB	TPC-H
$\|D\|$	48k	400k	100k - 3.4m
T_n	1	4	8
$\|Attr\|$	14	23	61

5.1.1 Datasets. We conducted the experiments on three real large datasets. Table 7 shows the basic information of the three datasets including the size of datasets($|D|$), the number of tables(T_n) and total number of attributes in the datasets ($|Attr|$).

TPC-H. TPC-H[2] is a set of test benchmarks for database decision support capability, which generates database tables and complex queries through the TPC-H tool. In this experiment, we use $1SF$ as a parameter to generate the database and divide the database size into four versions of 3.4m, 1m, 300k and 100k to test the algorithm.

Census Income. The Census Income [7] dataset was extracted from the 1994 Census database. It is used to predict whether income exceeds $50K/year$ based on census data. Also known as "Adult" dataset. We use Census Income to test the performance of the algorithms with small data sets, especially for quality evaluation.

IMDB. The IMDB total dataset contains 7 tables, and we use a subset of it including *akas*, *basics*, *crew* and *ratings*. We randomly select 400k tuples from the dataset mainly for our quality experiments.

5.1.2 Parameters. To make the summary as simple as possible, we set the number of rules in a summary $K = 4$. The upper bound (ubw) for the weight of the rules was set to 4, while the lower bound (lbw) was set to 2 in the *IPS* algorithm. According to K, we set the number of context rules no more than $K' = \frac{\lceil K \rceil}{4}$. Considering the

[2]https://www.tpc.org

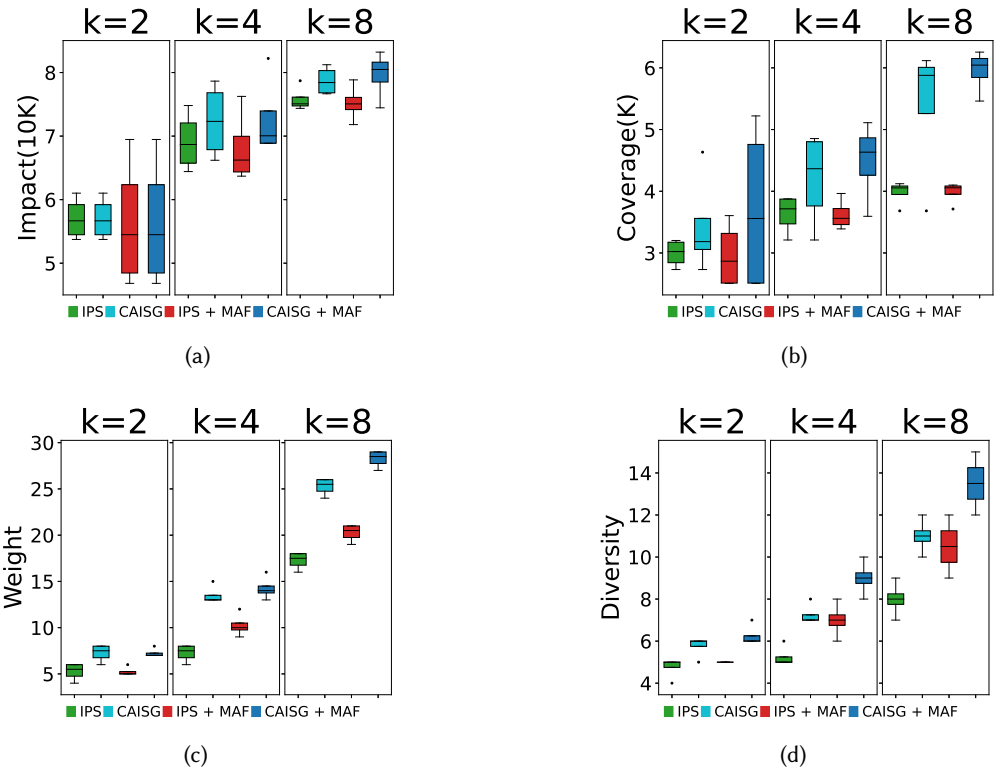


Figure 3: Quality metrics of summaries generated by four different algorithms on the Census Income dataset, (a) Impact; (b) Coverage; (c) Weight; (d) Diversity.

extreme attributes, the three thresholds used for attribute filtering in our algorithm are strictly set to $\lambda_{dispersive} = 50$, $\lambda_{object} = 0.9$, $\lambda_{dependent} = 0.95$.

5.2 Runtime Evaluation

To verify the effect of our two approaches on the runtime of the algorithms, we run four different combinations of the algorithms and record the runtime on all three datasets, including *IPS*, *CAISG*, *IPS* + *MAF*, and *CAISG* + *MAF*. the queries used for the three datasets are shown in Table 8.

Table 8: Queries used in the runtime experiments

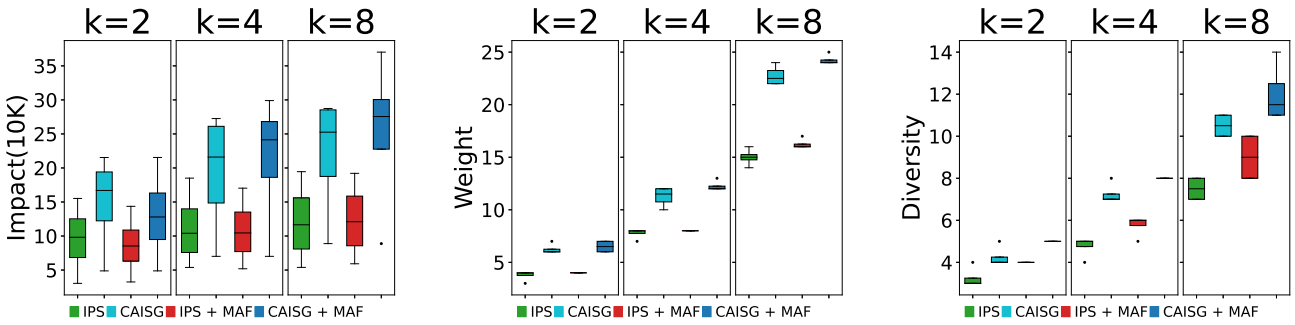
Dataset	Aggregate	GroupBy
TPC-H	quantity	returnflag, linestatus
IMDB	rating	genres
Census Income	fnlwgt	occupation

We first evaluate the effect of different database sizes on the *CAISG*+*MAF* algorithm on TPC-H. As shown in Fig. 1a, the runtime of the algorithm is positively correlated with the database size, and there is some difference in the runtime under different aggregate functions, which is mainly due to the different depth of mining the rules. Then we compared the difference in runtime of the four algorithms under three different aggregate functions, as shown in Fig. 1b, 1c, and 1d. As we expected, the *MAF* algorithm effectively reduces the algorithm runtime, while the *CAISG* algorithm takes the longest runtime due to the additional contextual information it computes.

In addition, we find that the *MAF* algorithm seems to offset the extra cost of the *CAISG* algorithm, and the cost of the *CAISG* + *MAF* algorithm is even less than that of the *IPS* algorithm. This fully verified that the *MAF* algorithm can effectively reduce the algorithm runtime, and showed similar results in the remaining two datasets, as shown in Fig. 2a and 2b. After the above experiments, it is obvious to conclude that the number of attributes has a much greater effect on the algorithm runtime than the database size, and the *MAF* algorithm can effectively filter the original attributes of the dataset.

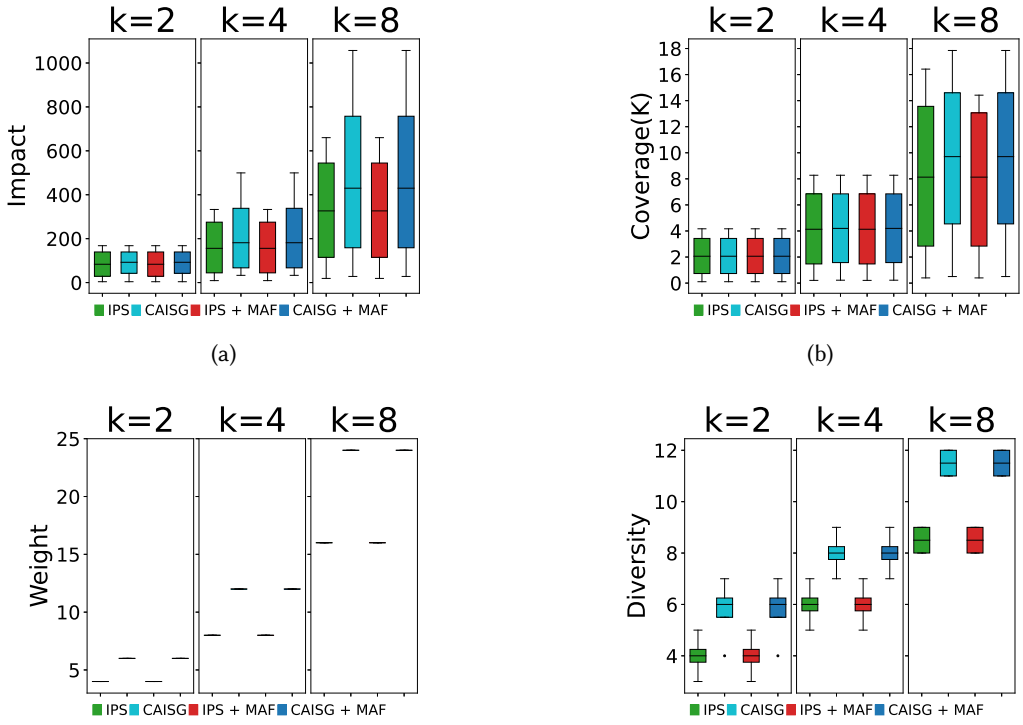
5.3 Quality Evaluation

Quality evaluation of summaries has been a longstanding challenge in the field, primarily due to its high subjectivity, posing significant challenges.In the experiment, besides employing four objective metrics to evaluate the quality of summaries, we also evaluated the feasibility of the annotation algorithm. In this section, we execute the same four algorithms on the full dataset and evaluate the quality of the summaries generated from all the query results, the queries used for the three datasets are shown in the Table 9. Most



(a) (b) (c)

Figure 4: Quality metrics of summaries generated by four different algorithms on the IMDB dataset, (a) Impact and Coverage(since the aggregate function is COUNT, Impact = Coverage); (c) Weight; (d) Diversity.



(a) (b) (c) (d)

Figure 5: Quality metrics of summaries generated by four different algorithms on the TPC-H dataset, (a) Impact; (b) Coverage; (c) Weight; (d) Diversity.

queries with high aggregation values are recorded in the experiment, ensuring the absence of randomness in our experimentation. It is worth noting that, for a more precise evaluation of summary quality metrics, our quality experiments will exclude attributes from user questions in the calculations.

In our experiment, four metrics for evaluating the quality of a summary are described. *Impact* represents the total impact of the rules in a summary. *Coverage* is the total number of tuples covered by the rules in a summary. Higher *Impact* and *Coverage* indicate that the summary includes a sufficient amount of high-value source metadata. Therefore, we analyze these two metrics together. *Weight* represents the sum weight of the rules in a summary, higher *Weight* means a better comprehensibility of a summary. *Diversity* is used to describe the number of unique attribute values in a summary,

Table 9: Queries used in the quality experiments

Dataset	AGG	Aggreate	GroupBy
TPC-H	SUM	tax	returnflag, linestatus
IMDB	COUNT	genres	genres
Census Income	AVG	fnlwgt	occupation

and we calculate it to evaluate the total information content of the summary. Furthermore, we will record the percentage of public features for all generated rules. While the size of this percentage is nearly meaningless, it is sufficient to demonstrate the effectiveness of our annotation algorithm.

5.3.1 Impact and Coverage. As shown in the Fig. 3a, 3b and 4a, as we expected the *CAISG* algorithm computes additional contextual information and replaces the low score rules, effectively improving the *Impact* and *Coverage* of the summary, especially at larger K values. While the *MAF* algorithm cannot improve them or even have a decrease at lower K values. This is because the dataset contains $Object$ attributes before the *MAF* algorithm is executed, resulting in unusually high *Impact* and *Coverage*, although $Object$ attributes can make the tuples more aggregated, but at the cost of greatly reducing interpretability. For example, the Census dataset is researched in the United States, and the majority of residents are of American nationality, so rules like $(United-States, male, \star, \star.....)$ are hardly informative, as verified in subsequent experiments. Unexpectedly, the *CAISG+MAF* algorithm is even more effective than the *CAISG* algorithm, probably because the *MAF* algorithm filters the $Object$ attributes to make the context information richer, and thus there is a better rule replacement.

The *MAF* algorithm seems to be ineffective for the TPC-H dataset (Fig. 5a and 5b). We tested and found that only $Dispersive$ attributes exist in the TPC-H dataset, and these attributes do not exist in the summary whether they are filtered or not, and at this time the *MAF* algorithm can only ensure to improve the efficiency of the algorithm while maintaining the same summary quality.

5.3.2 Weight and Diversity. As shown in the Fig. 3c , 3d , 4b , 4c , 5c , 5d , the *MAF* algorithm effectively improves the $Weight$ and $Diversity$, which proves that the total number of attributes is reduced but the number of attribute values appearing in the rules is larger and more diverse after the execution of the *MAF* algorithm. The *CAISG* algorithm also obtains higher $Weight$ and $Diversity$ with the additional attribute columns, and the improvement is higher than that of the *MAF* algorithm. The figures on TPC-H appear relatively clustered(Fig 5c, 5d). This phenomenon stems from TPC-H's data generation algorithm being uniform, resulting in concentrated weight and diversity metrics in the generated summaries.

In addition, we also explore whether both algorithms can be effective at the same time. Although the *MAF* algorithm affects the *CAISG* algorithm, this effect is usually positive, and it is clear from the execution results of the *CAISG+MAF* algorithm that their effects on $Weight$ and $Diversity$ can be effective simultaneously, even more than their sum.

5.3.3 Public Feature. Distinguishing feature rules as either public or private constitutes one of the most crucial aspects of our work. Therefore, it is imperative to evaluate the percentage of public features identified by the algorithm. As illustrated in Tables10 and 11, we have respectively documented the percentage of public features identified by the *CAISG* algorithm with and without the *MAF* algorithm. Clearly, our method can identify a certain number of public features across three datasets. This not only demonstrates the effectiveness of our approach but also confirms the existence of the problem we seek to address. The *MAF* algorithm has minimal impact on the identify of public features. An exception is observed in the *IMDB* dataset, where the '$isAdult$' object attribute is filtered out, leading to significant changes in the rules. However, aside from the extreme case of $K = 2$, the removal of object attributes has a positive effect on reducing the number of public features.

Table 10: The percentage of feature rules marked as public feature after running the CAISG algorithm.

Dataset	K=2	K=4	K=8
Census Income	5/8	7/16	13/32
IMDB	8/20	18/40	37/80
TPC-H	0/8	2/16	9/32

Table 11: The percentage of feature rules marked as public feature after running the CAISG+MAF algorithm.

Dataset	K=2	K=4	K=8
Census Income	3/8	6/16	14/32
IMDB	13/20	17/40	29/80
TPC-H	0/8	2/16	9/32

6 CONCLUSION

In this paper, we propose a novel provenance summary that augments the provenance summary using the context of the query result. Our approach provides crucial labels for provenance summaries, aiding users in selecting explanations more relevant to the problem at hand. We also propose an attribute filtering method for solving the problem of too many low-value attributes and inefficient generation of summaries. Our summaries are particularly suitable for explaining problems that have opposing meanings in the real world, and high-value explanations can always be obtained regardless of the user's previous knowledge of the data. We have conducted thorough efficiency and quality experiments on three datasets to prove the feasibility and effectiveness of our approach. Moreover, we unexpectedly discovered that the two methods we proposed seem to mutually optimize each other and yield results of higher value in certain circumstances.

ACKNOWLEDGMENTS

This work was supported by the Heilongjiang Key R&D Program of China under Grant No. GA23A915.

REFERENCES

[1] Zeinab Abbassi, Vahab S Mirrokni, and Mayur Thakur. 2013. Diversity maximization under matroid constraints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 32–40.

[2] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. 2009. Diversifying search results. In *Proceedings of the second ACM international conference on web search and data mining*. 5–14.

[3] Eleanor Ainy, Pierre Bourhis, Susan B Davidson, Daniel Deutch, and Tova Milo. 2015. Approximated summarization of data provenance. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 483–492.

[4] Omar AlOmeir, Eugenie Yujing Lai, Mostafa Milani, and Rachel Pottinger. 2021. Summarizing Provenance of Aggregate Query Results in Relational Databases. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 1955–1960.

[5] Bahareh Arab, Dieter Gawlick, Venkatesh Radhakrishnan, Hao Guo, and Boris Glavic. 2014. A generic provenance middleware for database queries, updates, and transactions.

[6] Bahareh Sadat Arab, Su Feng, Boris Glavic, Seokki Lee, Xing Niu, and Qitian Zeng. 2018. GProM-a swiss army knife for your provenance needs. *A Quarterly bulletin of the Computer Society of the IEEE Technical Committee on Data Engineering* 41, 1 (2018).

[7] Arthur Asuncion and David Newman. 2007. UCI machine learning repository.

[8] Nicole Bidoit, Melanie Herschel, and Katerina Tzompanaki. 2014. Query-based why-not provenance with nedexplain. In *Extending database technology (EDBT)*.

[9] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. 2001. Why and where: A characterization of data provenance. In *International conference on database theory*. Springer, 316–330.

[10] Peter Buneman and Wang-Chiew Tan. 2007. Provenance in databases. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*. 1171–1173.

[11] Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 335–336.

[12] Adriane Chapman and HV Jagadish. 2009. Why not?. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. 523–534.

[13] Adriane P Chapman, Hosagrahar V Jagadish, and Prakash Ramanan. 2008. Efficient provenance storage. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 993–1006.

[14] James Cheney, Laura Chiticariu, Wang-Chiew Tan, et al. 2009. Provenance in databases: Why, how, and where. *Foundations and Trends® in Databases* 1, 4 (2009), 379–474.

[15] Yingwei Cui and Jennifer Widom. 2000. Lineage tracing in a data warehousing system. In *Proceedings of 16th International Conference on Data Engineering (Cat. No. 00CB37073)*. IEEE, 683–684.

[16] Yingwei Cui and Jennifer Widom. 2000. Practical lineage tracing in data warehouses. In *Proceedings of 16th International Conference on Data Engineering (Cat. No. 00CB37073)*. IEEE, 367–378.

[17] Yingwei Cui, Jennifer Widom, and Janet L Wiener. 2000. Tracing the lineage of view data in a warehousing environment. *ACM Transactions on Database Systems (TODS)* 25, 2 (2000), 179–227.

[18] Ting Deng and Wenfei Fan. 2013. On the complexity of query result diversification. *Proceedings of the VLDB Endowment* 6, 8 (2013), 577–588.

[19] Marina Drosou and Evaggelia Pitoura. 2012. Disc diversity: result diversification based on dissimilarity and coverage. *arXiv preprint arXiv:1208.3533* (2012).

[20] Wenfei Fan, Xin Wang, and Yinghui Wu. 2013. Diversified top-k graph pattern matching. *Proceedings of the VLDB Endowment* 6, 13 (2013), 1510–1521.

[21] Piero Fraternali, Davide Martinenghi, and Marco Tagliasacchi. 2012. Top-k bounded diversification. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*. 421–432.

[22] Boris Glavic et al. 2021. Data provenance. *Foundations and Trends® in Databases* 9, 3-4 (2021), 209–441.

[23] Boris Glavic and Gustavo Alonso. 2009. Perm: Processing provenance and data on the same data model through query rewriting. In *2009 IEEE 25th International Conference on Data Engineering*. IEEE, 174–185.

[24] Todd J Green, Grigoris Karvounarakis, and Val Tannen. 2007. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 31–40.

[25] Manas Joglekar, Hector Garcia-Molina, and Aditya Parameswaran. 2015. Smart drill-down: A new data exploration operator. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, Vol. 8. NIH Public Access, 1928.

[26] Manas Joglekar, Hector Garcia-Molina, and Aditya Parameswaran. 2017. Interactive data exploration with smart drill-down. *IEEE Transactions on Knowledge and Data Engineering* 31, 1 (2017), 46–60.

[27] Bhargav Kanagal, Jian Li, and Amol Deshpande. 2011. Sensitivity analysis and explanations for robust query evaluation in probabilistic databases. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*. 841–852.

[28] Grigoris Karvounarakis, Zachary G Ives, and Val Tannen. 2010. Querying data provenance. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. 951–962.

[29] Seokki Lee, Bertram Ludäscher, and Boris Glavic. 2019. PUG: a framework and practical implementation for why and why-not provenance. *The VLDB Journal* 28, 1 (2019), 47–71.

[30] Seokki Lee, Bertram Ludäscher, and Boris Glavic. 2020. Approximate summaries for why and why-not provenance (extended version). *arXiv preprint arXiv:2002.00084* (2020).

[31] Seokki Lee, Xing Niu, Bertram Ludäscher, and Boris Glavic. 2017. Integrating Approximate Summarization with Provenance Capture.. In *TaPP*.

[32] Chenjie Li, Zhengjie Miao, Qitian Zeng, Boris Glavic, and Sudeepa Roy. 2021. Putting Things into Context: Rich Explanations for Query Answers using Join Graphs. In *Proceedings of the 2021 International Conference on Management of Data*. 1051–1063.

[33] Zhengjie Miao, Qitian Zeng, Boris Glavic, and Sudeepa Roy. 2019. Going beyond provenance: Explaining query answers with pattern-based counterbalances. In *Proceedings of the 2019 International Conference on Management of Data*. 485–502.

[34] Zhengjie Miao, Qitian Zeng, Chenjie Li, Boris Glavic, Oliver Kennedy, and Sudeepa Roy. 2019. CAPE: explaining outliers by counterbalancing. *Proceedings of the VLDB Endowment* 12, 12 (2019), 1806–1809.

[35] Fotis Psallidas and Eugene Wu. 2018. Smoke: Fine-grained lineage at interactive speed. *arXiv preprint arXiv:1801.07237* (2018).

[36] Christopher Ré and Dan Suciu. 2008. Approximate lineage for probabilistic databases. *Proceedings of the VLDB Endowment* 1, 1 (2008), 797–808.

[37] Sudeepa Roy, Laurel Orr, and Dan Suciu. 2015. Explaining query answers with explanation-ready databases. *Proceedings of the VLDB Endowment* 9, 4 (2015), 348–359.

[38] Sudeepa Roy and Dan Suciu. 2014. A formal approach to finding explanations for database queries. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1579–1590.

[39] K Selçuk Candan, Huiping Cao, Yan Qi, and Maria Luisa Sapino. 2009. Alphasum: size-constrained table summarization using value lattices. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*. 96–107.

[40] Pierre Senellart, Louis Jachiet, Silviu Maniu, and Yann Ramusat. 2018. Provsql: Provenance and probability management in postgresql. *Proceedings of the VLDB Endowment (PVLDB)* 11, 12 (2018), 2034–2037.

[41] Yuhao Wen, Xiaodan Zhu, Sudeepa Roy, and Jun Yang. 2018. Interactive summarization and exploration of top aggregate query answers. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, Vol. 11. NIH Public Access, 2196.

[42] Yuhao Wen, Xiaodan Zhu, Sudeepa Roy, and Jun Yang. 2018. Qagview: Interactively summarizing high-valued aggregate query answers. In *Proceedings of the 2018 International Conference on Management of Data*. 1709–1712.

[43] Allison Woodruff and Michael Stonebraker. 1997. Supporting fine-grained data lineage in a database visualization environment. In *Proceedings 13th International Conference on Data Engineering*. IEEE, 91–102.

[44] Eugene Wu and Samuel Madden. 2013. Scorpion: Explaining away outliers in aggregate queries. (2013).