Supporting the Analysis and Explanation of Dependencies through an LLM-Powered Tool

Loredana Caruccio[1,†], Tullio Pizzuti[1,*,†] and Giuseppe Polese[1,†]

[1]Department of Computer Science, University of Salerno, Via Giovanni Paolo II, 132, 84084 Fisciano, Salerno, Italy

Abstract

Small but informative summaries of a data source extracted from Data Profiling processes, also called metadata, are widely used in Data Preparation processes such as Data Quality and Data Integration. Among the best-known metadata are certainly the relationships between the values of two subsets of attributes, namely Functional Dependencies, together with their generalization called Relaxed Functional Dependencies. The discovery process of this type of metadata turns out to be an extremely difficult problem, given the exponentially sized search space. Despite this, there are several approaches in the literature to address the problem in an effective and efficient manner. However, two main issues have to be also considered in exploring discovery results. Firstly, each algorithm tends to store discovery results in a distinctive and not user-friendly format; and secondly, there is a lack of support for the interpretation of these relationships, which could bring new knowledge if appropriately analyzed. To address these issues, this paper introduces DEAL, an LLM-powered tool for effective functional dependency explanation, and single-column metadata visualization. It also includes JEF, which standardizes discovery output information without the need of customizing discovery algorithms. To demonstrate the effectiveness and the usability of the DEAL tool, we also discuss the result obtained through a task-oriented user study.

Keywords

Data Profiling, Functional Dependencies, Relaxed Functional Dependencies, Data Visualization, Large Language Models

1. Introduction

Analyzing manufacturing processes or developing recommendation systems are classic examples of how data analysis can lead to improvements in real-world domains. However, in order to extract the most value and the best knowledge from this frenetic flow of data, increasingly powerful analysis and knowledge discovery algorithms are required. In addition, storing high-frequency data requires limiting data quality checks that have a strong impact on storage performance. Since the value of the information extracted from the data depends largely on the quality of the data itself, improving them requires efforts in the data pre-processing step.

Through *Data Preparation* [1] tasks, analysts can improve the quality of data using transformation, cleaning, integration, and reduction techniques, and consequently improve the insights extracted and the processes that use them. *Data Profiling* [2] processes support Data Preparation operations through the extraction of summaries, but informative *metadata*. Single-column

SEBD 2025: 33rd Symposium On Advanced Database Systems, June 16–19, 2025, Ischia, Italy

*Corresponding author.

†These authors contributed equally.

lcaruccio@unisa.it (L. Caruccio); tpizzuti@unisa.it (T. Pizzuti); gpolese@unisa.it (G. Polese)

statistics, multiple-column correlations, and data dependencies are examples of profiling metadata [2]. In particular, Functional Dependencies (FDs) represent one of the most used metadata to support data quality tasks [3][4], and their generalized version, namely Relaxed Functional Dependencies (RFDs) [5], which are capable of tolerating errors and inaccuracies in the data.

The discovery of RFDs from data is a computationally expensive task [2], due to the exponential number of combinations of attributes and all possible similarity thresholds that can be defined over attributes [6]. A further problem is the analysis and the interpretation of the results. Getting the most qualitative insight from the dependencies or identifying the best result that can be exploited in a specific context may require the intervene of domain experts. Nevertheless, a huge quantity of dependencies, sometimes *spurious* (i.e., dependencies that do not have meaning in the real-world but are strictly related to the values of the data), can result from a discovery process. To overcome these issues, it is possible to exploit Large Language Models (LLMs), due to their ability to understand natural language, and their knowledge on a wide range of fields and topics. In fact, although they cannot replace the knowledge and experience of a domain expert, LLMs can provide initial insights, especially when they can be guided through the use of advanced prompting techniques, which allows for tailoring the interaction with the model according to specific objectives.

This work aims to study the extent to which LLMs are able to support non-expert domain-users in the interpretation of data and the dependencies. The DEAL tool (Dependencies Explanation with Advanced Language Models) is built to support users in analyzing both single-column metadata and dependencies, by also allowing for the interaction between a user and an LLM, to provide semantic meaning to dependencies. Furthermore, to allow a proper organization of discovery results, we propose the JEF (JSON dEpendencies Formatter) system with the aim of standardizing the results of discovery algorithms in a JSON format.

The remainder of the paper is organized as follows. Section 2 shows the approach in literature for data and metadata analysis tools, together with LLM-based systems and tools that support specific-domain data analysis. Section 3 presents the background notions related to FDs and RFDs. Instead, Section 4 describes the DEAL system, by introducing its architectures and functionalities, together with the prompt-engineering process used to interact with the LLM. Section 5 provides an experimental evaluation of the usage of LLM for the dependency interpretation through a user study, and finally Section 6 draws conclusions and possible future directions.

2. Related work

This section provides an overview of recent works that support data analysts in the interpretation of metadata discovery results, together with tools that exploit LLMs to empower analysis tasks.

Data and Metadata Analysis Tools. Concerning the Data Profiling research area, Metanome [7] provides a framework for the deployment of discovery algorithms, and its visual interface allows the storing of algorithms, datasets, and results. In addition, it can rank the discovery results according to different metrics and visualize them through different visualization techniques. Data Civilizer [8], instead, is a modular, big data management system designed to aid in the discovery of relevant data for specific user-oriented tasks. Among the various modules, we can find those oriented to data cleaning and workflow definition. In addition, it allows managing dataset updates by propagating changes to tasks in order to update user-defined discovery information. Exploiting particular types of RFDs, called Conditional Functional Dependencies (CFDs) [5], Data Auditor [9] aims to explore data quality and data semantics. Instead, in [10], a technique for ranking and visualizing dependencies holding on big and multimedia data is proposed. The authors introduce different novel visual metaphors specifically oriented for providing a unique visualization of RFD discovery results. They also highlight the advantages of the proposed visualization metaphors through a qualitative evaluation. Finally, in a data-stream scenario, Stradyvar [11] allows the visualization of different types of RFDs. In addition, it enables their comparison when executed in various runs and with possible changes in data. It also provides visual manipulation operators to dynamically compose and filter results.

LLM-powered Analysis Tools. Concerning the usage of LLMs for the interpretation of domain-specific data and information, different approaches have been provided in the literature. In general, LLMs make it possible to generate explanations without the need for task-specific training using pre-existing knowledge (zero-shot), or adapt to a new explanation context when exposed to a limited number of examples of a task during the training or adaptation phase (few-shot). For example, the FOLK [12] approach exploits LLMs for claim verification, generating logical and detailed explanations without requiring annotated evidence, combining first-order logic reasoning with contextual knowledge learned by LLMs. On the other hand, some studies have approached the problem of explaining information concerning the health of patients. In [13], MentaLLaMA is proposed for interpretable mental health analytics, trained on a dataset of social posts related to explanations generated using expert-designed few-shot prompts on ChatGPT. Then, strict automatic and human evaluations have been performed to ensure reliability. Some other studies have focused on explaining or interpreting ECG signals, or on trying to improve diagnoses. In [14], Explainable AI (XAI) techniques are combined with Multimodal LLMs in order to enhance ECG diagnostic interpretability. Instead, in [15], a Multimodal LLM approach employs an innovative contrastive learning framework that synchronizes ECG waveform data with textual reports. By using ChatGPT to generate the extraction code of the main conceptual features from raw ECG data, the approach used in [16] is able to perform classification and, then, generate meaningful conceptual level explanations for the outcomes. The use of LLMs for the explanation and interpretation of complex data and results, which require analysis by domain experts, has not only been exploited in the complex medical field. In fact, in order to support the study of students, ChatGPT has been evaluated in [17] in order to explain reaction mechanisms in organic chemistry, focusing on how these technologies can be integrated into the chemistry classroom. Instead, LogPrompt [18] is an approach that utilizes LLMs and advanced prompting techniques for automated log analysis in zero-shot scenarios, without the need for in-domain training. This improves performance and provides interpretive explanations for analysts, particularly in dynamic and data-limited environments. Finally, [19] proposed a hybrid method to generate natural language explanations for security risks in automation rules. To this end, the authors trained an LLM by leveraging domain-specific prompts, improving the understanding of risks in the IoT domain.

3. Preliminaries

Among the best-known profiling metadata, a Functional Dependency (FD) [20] identifies relationships between values of two attribute sets. Formally, given an relational instance r of a relation schema R defined over a set of attributes $attr(R)$, an FD over R is a statement $X \to Y$, with $X, Y \subseteq attr(R)$, holding on r iff for every pair of tuples (t_1, t_2) in r, whenever $t_1[X] = t_2[X]$, then $t_1[Y] = t_2[Y]$, where X and Y represent the Left-Hand-Side (LHS) and Right-Hand-Side (RHS) of the FD, respectively.

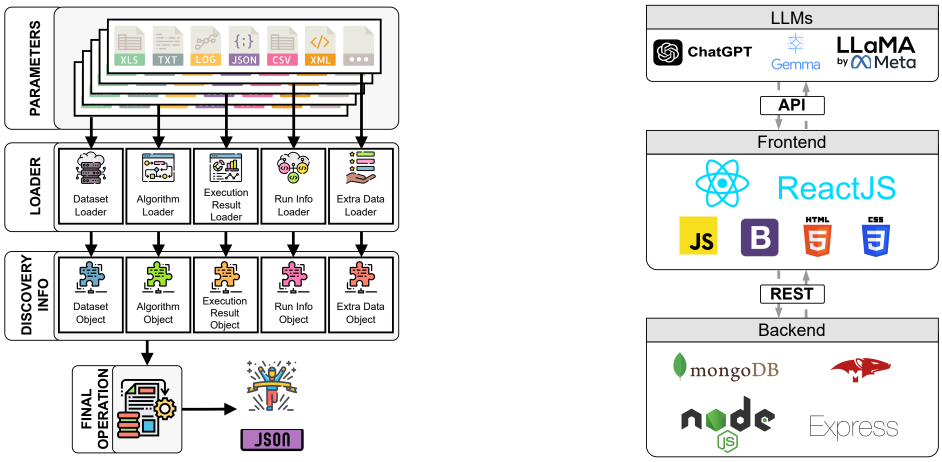
Due to the ever-increasing data subject to errors and inaccuracies, the equality constraint used for checking the attribute values and the constraint that requires the validity of the FD property on the entire dataset, turn out to be too restrictive. Over the year, the literature has extended (or relaxed) the concept of FD by defining different types of FDs, i.e., CFDs [21], DDs [22], AFDs [23], etc [5], and each of them can be used for different tasks to capture several types of relationship among data. This has led to the introduction of the general definition of Relaxed Functional Dependency (RFD) [5], which exploits distance or similarity functions for relaxing the "comparison operator" of attribute values, and/or relaxing the "extent", i.e., by using a coverage measure for identifying the amount of the tuples for which the dependency is valid. Based on the relaxation criteria, an RFD can be classified as RFD$_e$ (relaxing on the extent), RFD$_c$ (relaxing on attribute comparison), or hybrid RFD (relaxing on both).

Since the experimental evaluation mainly focuses on interpretation of RFD$_c$s, we introduce the formal definition RFD$_c$. Given a relation schema $R = (A_1, \ldots, A_m)$, an RFD φ on $\mathcal{R}$ is denoted by $X_{\Phi_1} \to Y_{\Phi_2}$, where X, Y are disjoint attribute subsets of $attr(R)$; a constraint Φ represents a conjunction of predicates, satisfied if $\forall(t_1, t_2)$ of an instance r of R, each similarity (or distance) constraint $\phi_i \in \Phi$ is satisfied. An RFD$_c$ can also be described in a compact way as $A_{\leq 0}, B_{\leq 0.5} \to C_{\leq 3}$, where A, B, C are attributes, θc represents the operator and the threshold used in a similarity (or distance) constraint $\delta\theta c$, where δ is a similarity or distance function, θ is a comparison operator ($<, \leq, \geq$, etc.), and c is a threshold. Examples of similarity (or distance) functions are Levenshtein distance [24] for textual value, i.e., $lev(s1, s2)$, and absolute distance for numerical value, i.e., $abs(x, y) = |x - y|$.

4. The proposed system

To the best of our knowledge, there is no standard storage format for dependencies, and each discovery algorithm tends to differently represent it. To address this issue, the DEAL tool provides a standard format which exploits the flexibility and extensibility of the JSON format. This representation is also used to store and manage discovery results through the non-relational DBMS MongoDB[1]. In fact, forming a standard that allows the results of dependency discovery algorithms to be stored not only guarantees simpler analysis and information extraction processes, but also makes it possible to avoid the necessity to re-run single algorithms for comparing experimental results. Indeed, profiling metadata discovery algorithms often requires considerable computational effort. Thus, a service that stores not only the metadata, but also execution times, errors, etc., would simplify future experiments and algorithm comparisons.

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



(a) JEF architecture used to organize dependencies discovery results into the JSON standard format.

(b) DEAL web tool architecture used to visualize and interact with the information contained in the JSON produced by JEF.

Figure 1: JEF and DEAL architectures.

4.1. JEF

In order to not impose a standard for formatting dependency discovery results to be associated with each possible algorithm, a Python[2] tool, named JEF, is provided. The aim JEF is to translate the discovery information by building a *DiscoveryInfo* object that contains information about the whole discovery process. The *DiscoveryInfo* is composed of five main objects: *(i) Dataset* containing information on the data storage format, header information, rows/columns dimensionality, and single-column statistical metadata; *(ii) Algorithm* containing general information about the executed algorithm, like name and programming language; *(iii) Execution result* containing the set of dependencies discovered by the algorithm and the relaxation criteria applied; the information of the time and memory used for the process; and any errors that have occurred; *(iv) Run info* containing information on the running parameters of execution and the system on which the algorithm was executed; and *(v) Extra data* optionally containing additional information. Thanks to *InfoLoader* classes, it is possible to construct the objects composing *DiscoveryInfo* and to process files generated by an algorithm into a JSON file. The *InfoLoader* class receives a map (<key, value> pairs) as a parameter, which will be then used by extended classes. In particular, to perform the translation, JEF requires the implementation of at least four specific classes: *DatasetLoader*, *AlgorithmLoader*, *ExecutionResultLoader*, and *RunInfoLoader*. Instead, the classes *ExtraInformationLoader* and the *FinalOperation* can be optionally implemented according to specific output formats of analyzed algorithms, such as the presence of RFD_cs without any specified LHS, meaning that all attributes can determine the RHS. The above-mentioned classes are used by the *Loader* class, which is responsible for building the object *DiscoveryInfo*.

Figure 1a shows this architecture, which not only facilitates the implementation of a translator,

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

Figure 2: Main views of the DEAL web tool.

but also allows for a strong reuse of the *InfoLoader* components. In fact, through JEF, it is possible to reuse already provided InfoLoaders, e.g., the *CSVDatasetLoader* class, which takes as input the dataset in a ".csv" format and extracts the information to define the *Dataset* object.

4.2. DEAL

The Web tool DEAL manages and analyzes JSON files representing *DiscoveryInfo* objects. As shown in Figure 1b, DEAL leverages the Web Framework ReactJS[3] to define visual interfaces that guarantee simple and proper user interactions. Then, the communication with the backend exploits REST services that enable CRUD operations on discovery result files. Finally, the explanation of discovery results is performed on the client-side through LLM APIs. All functionalities of the DEAL tool are exposed in *File management* and the *Data visualization* Web Pages.

4.2.1. File management

This Web page (Figure 2a) is responsible for the organization and management (Delete, Rename, Search, and Sort) of JSON files uploaded in DEAL. Moreover, it allows access to *DiscoveryInfo* data contained in the selected JSON file, providing access to the *Data visualization* page.

4.2.2. Data visualization

This Web page visualizes the information contained in a JSON file and enables interaction with LLMs in order to explain the semantic meaning of the discovered dependencies. In particular, among the other information, it is possible to see and analyze the execution information and discovery requirements, i.e., times and memory used (see Figure 2c) and the characteristics of the *Dataset* under analysis (see Figure 2b). In fact, thanks to the *DatasetLoader*, described in

[3] https://react.dev/

Section 4.1, different single-column metadata [2] are extracted and presented. The pre-defined *CSVDatasetLoader* class identifies the file dimension, the header, the column separator, the quantity of null values, the data domains, and the row/column dimensionality.

The next area, *Column Statistics*, allows users to become familiar with the data (see Figures 2b and 2d). In fact, for each column, Box Plots show the distribution of the values, highlighting the median, quartiles, ranges, and potential outliers. This visual representation helps users quickly grasp the variability, the mean, and the median of values, and the presence of anomalies within each column. Then, while Min/Max and Top-K Pie Charts can be used for a high-level data exploration, the Null Values chart can be used for the identification of quality issues in the dataset [2]. Instead, the *Algorithm* section provides useful data concerning the execution parameters and the runtime information, such as execution time or possible errors that happened during the discovery. The *Dependency analysis* section aims to provide insights about LHSs, RHSs, and possible thresholds of the attributes that characterize the discovered dependencies. In particular, these charts show the amount of dependencies with respect to LHS cardinality, the frequencies in which an attribute appears on each side according to all possible distance thresholds included in the results (see Figure 2e), and the list of all attributes implying each specific RHS. It is worth noting that by interacting with the attributes in the header and the different charts, it is possible to accordingly filter the dependencies, thus allowing a user to focus only on the subset of results that s/he considers as the most meaningful one. Finally, a user can select the dependencies for which s/he needs an accurate explanation, and by interacting with pre-configured prompts, s/he can request an LLM to interpret them (see Figure 2f). More specifically, two pre-configured prompts are provided, the first requires the simple interpretation of dependencies by taking into account only the attribute names, while the second also considers single-column metadata to provide enhanced information to the LLMs. The user can also edit the prompt in order to require specific information to be considered during the explanation, and/or request a *summary* when the explanation appears to be long and dispersive.

4.2.3. Prompting stategy

To obtain relevant and accurate answers from an LLM, it is necessary to make use of prompts that can guide the interaction in a clear and structured manner. Effective prompts must contain specific and detailed requests, providing a precise objective to the model and introducing the context in a clear and simple manner. Identifying a prompt that allows the semantic interpretation of dependencies, by also making use of data values and single-column profiling metadata, has required an iterative optimization process, in which the analysis of responses, the identification of areas for improvements, and the manual adaptation of prompts were key steps.

In fact, to obtain the final prompt, which generally allows for accurate responses from LLMs, a simple prompt is iteratively refined: *Explain the functional dependency* $A \rightarrow B$. Then, the concept of similarity threshold is introduced in the RFD representation, i.e., *Explain the relaxed functional dependency* $A@thr_a \rightarrow B@thr_b$, in order to allow the model to identify the context of RFD. The model used for the prompt refinement is ChatGPT based on GPT-4-turbo, which was able to automatically identify the context and link the thresholds to the relaxation on the comparison of attribute values. Subsequently, the specific objective of explaining dependency in depth was added to the prompt, as well as a more descriptive definition of RFDs. This could allow

I would like a thorough understanding of the RFD (Relaxed Functional Dependency) dependencies listed below, holding on [Dataset name] dataset. An RFD is a relationship between variables where a set of attributes (lhs - left-hand side) determines another attribute (rhs - right-hand side), with specific tolerance thresholds indicated. The notation for an RFD is structured as follows: $attribute_x@thr_x, attribute_y@thr_y, \cdots \rightarrow attribute_z@thr_z$ where for each attribute the tolerance threshold used to compare the similarity of the attribute values is indicated after the @ character. Provide an explanation of each attribute and dependency, based on the attribute names and associated tolerance thresholds, delving into the semantic aspects of how these attributes interact. Describe how this information affects these relationships and how the semantic meaning of the attribute names characterizes the dependency in the real world. The dependencies are as follows: [List of selected dependencies]

Figure 3: Final version of the pre-defined prompt for the interpretation of dependencies.

for more precise responses, but possibly too long, especially in the case of dependencies with many attributes. Thus, to delve into the semantic aspects of the correlation, the final formulation of the prompt allowed for having the formal definition RFD in the prompt, but by introducing an explanation of attributes and thresholds included in the LHS and RHS of the dependency, as shown in Figure 3. Finally, in order to provide a demonstration of the effectiveness of the prompt, in Figure 4, an FD extracted from the dataset Wine Quality[4] is explained through the specialized prompt using ChatGPT (GPT-4-turbo). This dataset contains several physicochemical properties of wine samples, such as acidity, sugar content, sulfur dioxide levels, and alcohol percentage. These attributes, generally, are used to predict the quality of a wine, which human tasters score.

5. Experimental evaluation

To show how LLMs can help users to enrich their knowledge about data and discovered dependencies, we designed and conducted a user study involving 15 experienced Data Scientists with no specific domain expertise. In fact, we choose the dataset “Pittsburgh Bridges”[5], containing information about 108 bridges built in Pittsburgh since 1818 since the interpretation of dependencies does not require specific knowledge of the domain. Furthermore, it contains data of different kinds, such as integer and decimal, categorical and string data, and several missing values, resulting in a dataset that is also difficult to interpret from a Data Profiling perspective. The main attributes of this dataset are: RIVER - the river it crosses; LOCATION - the geographical position within the study area; ERECTED - the year of its construction; PURPOSE - its primary function (i.e., used for vehicles, trains, pedestrians, or other purposes); LENGTH - its total length; LANES - the number of available lanes; T-OR-D - if traffic passes above or through the supporting framework; CLEAR-G - the clear height under the bridge; MATERIAL - the main material used for the construction; SPAN - the distance between its support piers; REL-L - its relative length; and TYPE - the structural design based on its architectural configuration.

For the discovery of RFD, the algorithm DiMε [25] is used, and to identify a specific threshold for each attribute, the distance distribution between each pair of values has been evaluated. Consequently, the thresholds have been set to allow for the selection of a proper quantity of similar values with respect to the total number of tuple pairs. For categorical attributes, the comparison thresholds were left at 0, and for the others they were defined as follows: LOCATION=0.2, ERECTED=10, LENGTH=5, LANES=1, and REL-L=1.

[4]https://archive.ics.uci.edu/dataset/186/wine+quality
[5]https://archive.ics.uci.edu/dataset/18/pittsburgh+bridges

To thoroughly understand the Relaxed Functional Dependency (RFD) fixed_acidity@0.00, chlorides@0.00, total_sulfur_dioxide@0.00, density@0.00 -> quality@0.00 in the Wine Quality dataset, let's break it down step by step:

1. Understanding the Attributes

fixed_acidity@0.00: *Definition:* The fixed acidity in wine, typically associated with acids like tartaric, malic, and citric acid. *Tolerance:* 0.00 means that the value for fixed acidity must be exactly the same between two records. No variation is allowed. *Semantic Role:* Fixed acidity influences the taste of wine, contributing to its crispness and freshness. It also impacts the overall stability and preservation of the wine.

chlorides@0.00: *Definition:* The level of chlorides (salt) in wine, which can affect both the taste and the preservation of the wine. *Tolerance:* 0.00 means that the chloride content must match exactly between two records. Any variation is not tolerated. *Semantic Role:* Chlorides can influence the flavor of wine, and higher levels can lead to a salty taste. Consistency in chloride content is important for the wine's overall flavor profile.

total_sulfur_dioxide@0.00: *Definition:* The amount of sulfur dioxide present in the wine, used as a preservative and to prevent oxidation. *Tolerance:* 0.00 means no variation is allowed in the sulfur dioxide content. *Semantic Role:* Sulfur dioxide affects the wine's preservation and its ability to age well. It also influences the aroma and taste, especially in terms of freshness.

density@0.00: *Definition:* The density of the wine, which is a measure of its mass per unit volume. Density is influenced by the alcohol and sugar content. *Tolerance:* 0.00 means that the density must be exactly the same between two records. *Semantic Role:* Density can provide insights into the sugar content and alcohol concentration in the wine, both of which affect the wine's mouthfeel and body.

2. Semantic Interpretation of the RFD

The RFD states that: If two wine records have exactly the same values for fixed_acidity, chlorides, total_sulfur_dioxide, and density (no tolerance allowed), then they must have the same value for quality. *In other words:* If two wines are identical in terms of their fixed acidity, chlorides, sulfur dioxide, and density levels, they must have the same quality rating.

These attributes (acidity, chloride content, sulfur dioxide, and density) are critical factors that contribute to the overall quality of the wine. A wine's quality, as rated in the dataset, is strongly influenced by its chemical composition. Since there is zero tolerance for any variation in these attributes, any small differences would lead to a mismatch in quality. This RFD implies that small changes in these chemical factors will directly affect the wine's perceived quality.

3. Real-World Implications

Consistency in Wine Quality Rating: This RFD ensures that wines with the exact same levels of acidity, chlorides, sulfur dioxide, and density are rated the same in terms of quality. This adds consistency and reliability to the dataset, ensuring that quality ratings are truly reflective of the chemical composition of the wine. *Error Detection:* If two wines with identical levels of acidity, chlorides, sulfur dioxide, and density are rated differently in terms of quality, it could indicate an error in the dataset or an inconsistency in the quality rating process. *Tolerance Justification:* The 0.00 tolerance for all attributes (fixed_acidity, chlorides, total_sulfur_dioxide, density) reflects the idea that small variations in these factors (such as slight changes in acidity or chloride content) should not lead to different quality ratings. If these chemical attributes are identical, the wine quality rating should be identical as well.

4. Conclusion

This RFD illustrates that chemical attributes such as fixed acidity, chlorides, total sulfur dioxide, and density are strongly tied to the quality of the wine. By enforcing zero tolerance for variation in these attributes, it ensures that wines with identical chemical compositions are consistently rated the same in terms of quality. This RFD helps maintain geographical or production consistency in the dataset by aligning wine quality with its chemical profile. If two wines share the same chemical characteristics but differ in quality, this may indicate a problem with the quality rating system or an error in data collection. In practice, this RFD ensures accuracy in quality prediction based on the fundamental components that shape the wine's characteristics and overall taste profile.

Figure 4: Interpretation of an FD extracted from the Wine Quality dataset.

Without the use of DEAL, a user would have to manually analyze the dataset features, subsequently constructing graphs and box plots, and extracting single-column statistics. Filtering attributes and data are difficult tasks to do manually; in fact, each filter has an impact on all charts, information, and dependencies extracted. After that, a user must build the prompt and run it on the various tools provided by the developers to interact with LLMs. The DEAL tool, already provides the appropriate JEF translation classes to make compatible dependencies extracted with DIMε. Once the dependencies have been translated, uploading the JSON file in DEAL allows visualization through interactive charts of single-column metadata. A user can filter attributes and thresholds through the appropriate labels shown over each chart, and each change is reflected in the overall information on the page, including dependencies. In this way, a user can focus on searching for intrinsic knowledge in the data without worrying about having to refresh the information. Finally, thanks to an LLM, a user can generate clear and understandable explanations of selected dependencies. Although there is the possibility of changing them, the default prompts can guide LLMs in explaining the RFD(s), thus allowing the user to interpret the results simply by interacting with the Web interface.

To assess the ability of an LLM, i.e., ChatGPT based on GPT-4-turbo, in the interpretation of

Table 1
Meaning of quantitative scores provided by users by means of a Likert Scale.

Score	Metrics	Description
1/5	Accuracy	The answer misunderstands the context or ignores key concepts asked
	Specificity	The answer is generic and adds no value beyond a superficial definition.
	Fluency	The answer is confusing, disorganized, or difficult to follow.
2/5	Accuracy	The answer contains significant errors in the interpretation of the prompt or does not adequately capture the context. Some key concepts are incorrect or are missing.
	Specificity	The answer offers only a superficial view of the concept, with limited information and few connections.
	Fluency	The answer is difficult to understand due to structural problems or ambiguous language.
3/5	Accuracy	The answer captures the context but contains inaccuracies regarding key details.
	Specificity	The answer is clear but lacks important details or meaningful examples.
	Fluency	The answer is understandable, but there are gaps in structure or organization.
4/5	Accuracy	The answer is correct and consistent with the prompt, with a good understanding of the context and the concepts requested, although with small imperfections.
	Specificity	The answer is detailed but could include further examples or clarification.
	Fluency	The answer is clear, but it could be improved in terms of organization or simplification.
5/5	Accuracy	The answer perfectly interprets the prompt, completely captures context, and responds accurately, coherently, and completely.
	Specificity	The answer explores the concept fully, offers concrete examples and reflects on the implications.
	Fluency	The answer is well-structured, clear, and understandable.

Table 2
Dependencies used for the experimental evaluation of the LLM's interpretability.

Id.	Dependency
φ_1	$\mathsf{LOCATION}_{\leq 0.20} \rightarrow \mathsf{RIVER}_{\leq 0}$
φ_2	$\mathsf{LOCATION}_{\leq 0.20}, \mathsf{ERECTED}_{\leq 10}, \mathsf{TYPE}_{\leq 0} \rightarrow \mathsf{MATERIAL}_{\leq 0}$
φ_3	$\mathsf{LOCATION}_{\leq 0.20}, \mathsf{LENGTH}_{\leq 5} \rightarrow \mathsf{TYPE}_{\leq 0}$

dependencies, we asked users to use the DEAL tool to analyze a specific subset of RFDs holding on the Pittsburgh Bridges dataset. After asking each user to familiarise themselves with the dataset and the metadata displayed through DEAL, they each completed specific dependency filtering tasks so that to obtain the same subset of RFDs (Table 2). In particular, the dependency φ_1 indicates that the nearest bridges should cross the same river, and φ_2 states that for bridges built in the same area, with a 10-year build range, and the same structural design, the same material is used. In other words, this could indicate that the type of climate for the area and/or particular styles of bridge construction, also dictated by the standards of a particular period, require specific types of material to be built. Finally, φ_3 indicates that the nearest bridges with a similar length require a specific structural design or architectural configuration.

After the request for interpretation of a single dependency by the LLM, each user has assigned a score, from 1 to 5 (i.e., by following the Likert scale shown in Table 1), to three evaluation metrics: *i) Accuracy* that evaluates the model's ability to correctly interpret the prompt provided,

To thoroughly understand the Relaxed Functional Dependency (RFD) LOCATION@0.20 -> RIVER@0.00 on the bridges.csv dataset, let's break it down step by step.
1. Understanding the Attributes
LOCATION: This attribute likely refers to a geographical location where a bridge is situated. It could be represented as a city, town, region, or specific coordinates (latitude/longitude). The tolerance 0.20 suggests that similar but not necessarily identical values are considered equivalent. This means small variations in location names or geographical proximity may be allowed.
RIVER: This attribute probably indicates the river that the bridge crosses. It might be represented by river names (e.g., "Mississippi", "Amazon", "Nile"). The tolerance 0.00 means no variation is allowed, meaning that the river names must be an exact match.
2. Semantic Interpretation of the RFD
The RFD states that: If two bridge records have similar LOCATION values (within a 0.20 tolerance), then they must have exactly the same RIVER value.
In other words: Bridges in roughly the same geographical area should always cross the same river. Small differences in the LOCATION attribute (e.g., "New York City" vs. "NYC" or slightly different GPS coordinates) do not matter, but they should always refer to the same river. This implies that a bridge's geographical placement determines the river it crosses.
3. Real-World Implications
Geospatial Consistency: This rule enforces consistency in the dataset by ensuring that if two bridges are considered "close enough" in location, they must be recorded over the same river. *Error Detection:* If a bridge at the same approximate location is associated with different rivers, it could indicate an error in the data (e.g., incorrect river assignment). *Tolerance Justification:* The 0.20 threshold for LOCATION accounts for slight variations in naming conventions (e.g., "Paris" vs. "Paris, France") or minor differences in coordinate rounding. The 0.00 threshold for RIVER ensures that rivers are uniquely identified and prevents inconsistencies (e.g., a bridge recorded as crossing both the "Amazon" and "Rio Negro" mistakenly).
4. Conclusion
This RFD ensures that bridges in roughly the same location must cross the same river, allowing for small variations in how the location is recorded but requiring exact consistency for river names. It supports data consistency, helps detect errors, and reflects a real-world geographic constraint in the dataset.

Figure 5: Interpretation of an RFD extracted from the Bridges dataset.

capture the specific context based on the information provided, and respond consistently with the requested key concepts; *ii) Specificity* that evaluates the level of detail and the ability of the model to go beyond superficial explanation, including critical analysis, theoretical implications, and practical applications; and *iii) Fluency* that evaluates the readability, organization, and understandability of the response.

Notice that the JEF and DEAL source code, documentation, and all resources related to the study are available on GitHub[6].

5.1. Results evaluation & Discussion

An example of interpretation of the RFD: $\mathsf{LOCATION}_{\leq 0.20} \rightarrow \mathsf{RIVER}_{\leq 0}$ provided by ChatGPT based on GPT-4-turbo (i.e., the model used in the user study) is shown in Figure 5. As it can be seen from the answer, the model can perfectly understand the context of the dataset, the meaning of the attributes, and the meaning of the RFD. Furthermore, it is capable of understanding neighbouring positions, since they are considered equivalent with a 0.20 tolerance threshold, whereas the 0.00 threshold indicates the exact match of RIVER values. Finally, it also highlights the geospatial implications and a practical application in the error detection of this information. By providing examples of values, and proposing a justification for the relaxation on the LOCATION attribute, the LLM managed to cover the interpretation of dependency in a complete and comprehensive manner, even though the example also contains some errors because it incorrectly interpreted the value 0.20 as a percentage.

As described above, the participants analysed and scored the LLM responses with respect to the three described metrics. The results of the experiment are shown in Figure 6, in which the Box Plots of the response distribution for each metric are shown. As can be seen from the chart, LLM responses received high scores for all metrics analyzed, having a more consistent distribution of scores between values 4 and 5. While Accuracy and Specificity maintain the same mean, median, and the same distribution of outliers, the Fluency rating is slightly lower.

[6]https://github.com/DastLab/DEAL

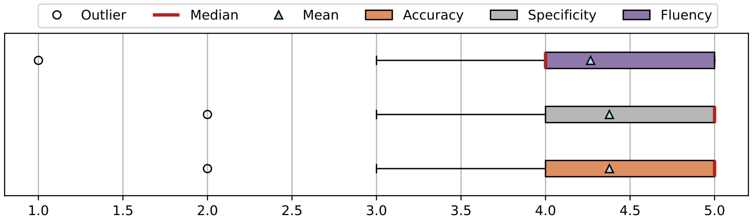


Figure 6: Distribution of cumulative scores, attributed by data scientists, for each of the metrics.

This may be due to the fact that this metric is more affected by subjective consideration with respect to the other ones. In fact, as also shown by the example of the answer, a user may positively evaluate the organization of the answer in points, while another may prefer to have a more discursive explanation. Furthermore, it must be taken into account that each query of the model, even with the same prompt, leads to different answers. Finally, users not only positively evaluated the application of LLMs in explaining dependencies, but also the simplicity and friendliness of using a tool like DEAL to analyse data and metadata.

6. Conclusion & Future work

With the aim of analyzing the effectiveness of providing semantic interpretation of dependency discovery results, in this study, we proposed the LLM-powered tool DEAL. Interpret and grasping the impact of these relationships requires specific knowledge in the data domain itself. Thanks to LLMs, which provides knowledge with respect to a wide range of topics. In spite of this, an LLM cannot be completely relied upon, especially in activities that are critical or present security risks. The tool DEAL can also support the analysis of single-column metadata and dependencies discovered from a dataset through the interaction with visual components together with filtering, ranking, and selection functionalities. In addition, it also aims to define a standard in formatting discovery results.

In the future, we would like to extend the DEAL's functionalities, allowing for full modularity and compatibility with different metadata, in order to produce a complete analysis environment. Moreover, we are interested in comparing explanations of several LLMs on other types of metadata and/or domain-related information.

Acknowledgments

This work was partially supported by the PNRR MUR project PE0000013-FAIR (Future Artificial Intelligence Research).

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] A. Kochański, Data preparation, Computer Methods in Materials Science 10 (2010) 25–29.

[2] Z. Abedjan, L. Golab, F. Naumann, T. Papenbrock, Data profiling, volume 10, Springer, 2019.

[3] S. Kandel, R. Parikh, A. Paepcke, J. M. Hellerstein, J. Heer, Profiler: integrated statistical analysis and visualization for data quality assessment, in: Proceedings of the International Working Conference on Advanced Visual Interfaces, 2012, p. 547–554.

[4] L. L. Pipino, Y. W. Lee, R. Y. Wang, Data quality assessment, Communications of the ACM 45 (2002) 211–218.

[5] L. Caruccio, V. Deufemia, G. Polese, Relaxed functional dependencies—a survey of approaches, IEEE Transactions on Knowledge and Data Engineering 28 (2015) 147–165.

[6] T. Papenbrock, J. Ehrlich, J. Marten, T. Neubert, J.-P. Rudolph, M. Schönberg, J. Zwiener, F. Naumann, Functional dependency discovery: An experimental evaluation of seven algorithms, Proceedings of the VLDB Endowment 8 (2015) 1082–1093.

[7] T. Papenbrock, T. Bergmann, M. Finke, J. Zwiener, F. Naumann, Data profiling with metanome, Proceedings of the VLDB Endowment 8 (2015) 1860–1863.

[8] R. Castro Fernandez, D. Deng, E. Mansour, A. A. Qahtan, W. Tao, Z. Abedjan, A. Elmagarmid, I. F. Ilyas, S. Madden, M. Ouzzani, et al., A demo of the data civilizer system, in: Proceedings of the 2017 ACM International Conference on Management of Data, 2017, pp. 1639–1642.

[9] L. Golab, H. Karloff, F. Korn, D. Srivastava, Data auditor: exploring data quality and semantics using pattern tableaux, Proc. VLDB Endow. 3 (2010) 1641–1644.

[10] L. Caruccio, V. Deufemia, G. Polese, Visualization of (multimedia) dependencies from big data, Multimedia Tools and Applications 78 (2019) 33151–33167.

[11] B. Breve, L. Caruccio, S. Cirillo, V. Deufemia, G. Polese, Dependency visualization in data stream profiling, Big Data Research 25 (2021) 100240.

[12] H. Wang, K. Shu, Explainable claim verification via knowledge-grounded reasoning with large language models, in: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023, pp. 6288–6304.

[13] K. Yang, T. Zhang, Z. Kuang, Q. Xie, J. Huang, S. Ananiadou, Mentallama: interpretable mental health analysis on social media with large language models, in: Proceedings of the ACM Web Conference 2024, 2024, pp. 4489–4500.

[14] S. Wu, J. Zhou, Y. Dong, F. Chen, Enhancing explainability of deep learning-based ecg diagnosis using large language models, in: Proceedings of the 2024 8th International Conference on Advances in Artificial Intelligence, 2025, p. 61–65.

[15] M. Guo, Y. Zhou, S. Tang, Multimodal models for comprehensive cardiac diagnostics via ecg interpretation, in: Proceedings of the 2024 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), 2024, pp. 5756–5763.

[16] A. Sangroya, S. Jain, C. Anantaram, L. Vig, Conceptual explanations of ecg classification using large language models, in: Proceedings of the Third International Conference on AI-ML Systems, 2023, pp. 1–3.

[17] B. J. Yik, A. J. Dood, Chatgpt convincingly explains organic chemistry reaction mechanisms slightly inaccurately with high levels of explanation sophistication, Journal of Chemical Education 101 (2024) 1836–1846.

[18] Y. Liu, S. Tao, W. Meng, F. Yao, X. Zhao, H. Yang, Logprompt: Prompt engineering towards zero-shot and interpretable log analysis, in: Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings, 2024, p. 364–365.
[19] B. Breve, G. Cimino, V. Deufemia, Hybrid prompt learning for generating justifications of security risks in automation rules, ACM Trans. Intell. Syst. Technol. 15 (2024).
[20] E. F. Codd, Further normalization of the data base relational model, Data base systems 6 (1972) 33–64.
[21] W. Fan, F. Geerts, X. Jia, A. Kementsietsidis, Conditional functional dependencies for capturing data inconsistencies, ACM Transactions on Database Systems (TODS) 33 (2008) 1–48.
[22] S. Song, L. Chen, Differential dependencies: Reasoning and discovery, ACM Transactions on Database Systems (TODS) 36 (2011) 1–41.
[23] R. S. King, J. J. Legendre, Discovery of functional and approximate functional dependencies in relational databases, Journal of Applied Mathematics and Decision Sciences 7 (2003) 49–59.
[24] V. I. Levenshtein, et al., Binary codes capable of correcting deletions, insertions, and reversals, in: Soviet physics doklady, 1966, pp. 707–710.
[25] L. Caruccio, V. Deufemia, G. Polese, Mining relaxed functional dependencies from data, Data Mining and Knowledge Discovery 34 (2020) 443–477.
[26] J. B. Longwell, I. Hirsch, F. Binder, G. A. G. Conchas, D. Mau, R. Jang, R. G. Krishnan, R. C. Grant, Performance of large language models on medical oncology examination questions, JAMA Netw Open 7 (2024) e2417641.
[27] J. Kivinen, H. Mannila, Approximate inference of functional dependencies from relations, Theoretical Computer Science 149 (1995) 129–149.