EvoVis: A Visual Analytics Method to Understand the Labeling Iterations in Data Programming

Sisi Li, Guanzhong Liu, Tianxiang Wei, Shichao Jia, Jiawan Zhang*

Abstract—Obtaining high-quality labeled training data poses a significant bottleneck in the domain of machine learning. Data programming has emerged as a new paradigm to address this issue by converting human knowledge into labeling functions(LFs) to quickly produce low-cost probabilistic labels. To ensure the quality of labeled data, data programmers commonly iterate LFs for many rounds until satisfactory performance is achieved. However, the challenge in understanding the labeling iterations stems from interpreting the intricate relationships between data programming elements, exacerbated by their many-to-many and directed characteristics, inconsistent formats, and the large scale of data typically involved in labeling tasks. These complexities may impede the evaluation of label quality, identification of areas for improvement, and the effective optimization of LFs for acquiring high-quality labeled data. In this paper, we introduce EvoVis, a visual analytics method for multi-class text labeling tasks. It seamlessly integrates relationship analysis and temporal overview to display contextual and historical information on a single screen, aiding in explaining the labeling iterations in data programming. We assessed its utility and effectiveness through case studies and user studies. The results indicate that EvoVis can effectively assist data programmers in understanding labeling iterations and improving the quality of labeled data, as evidenced by an increase of 0.16 in the average F1 score when compared to the default analysis tool.

Index Terms—Visual analytics, model interpretation, data programming, data labeling

1 Introduction

Data programming is a powerful technique for acquiring labeled data through programmatically encoding expert knowledge to assign probabilistic labels to specific subsets of datasets. An example of data programming workflow in a text labeling task is illustrated in Fig. 1. In this process, domain experts initially create labeling functions (LFs) based on their task-specific expertise, which are then applied to the target dataset to create an applied matrix of weak labels. An automated label model is trained from this matrix and subsequently generates probabilistic labels with uncertainty for the collected dataset. Compared to traditional label methods [1], data programming offers greater efficiency, scalability, and flexibility [2]. Owing to its potential ability to address challenging labeling problems [3], data programming has garnered significant attention from natural language processing [4], computer vision [5], and multimedia processing communities [6].

As the mechanism of data programming typically involves training an ML label model, which aligns with the best practices [7] for training ML models including optimization, evaluation, and iterative improvement to achieve optimal model performance, the iterative optimization of the label model stands as an indispensable step in enhancing the quality of labeled data. Thus, interpreting the process holds important value as it further enhances labeling by fostering collaboration among data programmers, facilitating the optimization of LFs, and boosting the confidence of decision-making, which helps to harness the full potential of data programming by effectively combining human expertise with the power of automated labeling.

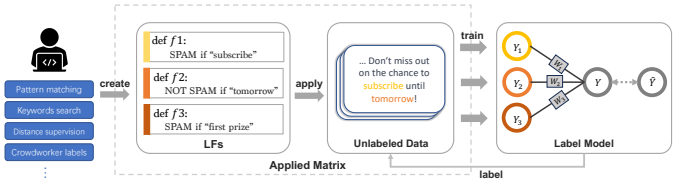


Fig. 1. The workflow of data programming for text labeling tasks. First, domain experts engage in the development of a set of LFs grounded in various heuristic methods; Second, these LFs are applied to each sample within the target dataset, resulting in the creation of an applied matrix consisting of weak labels; Third, a label model is trained autonomously, learning the weights associated with each LF from this applied matrix and text. Finally, the trained label model is employed in the collected dataset, generating probabilistic labels for samples.

However, the opacity and complexity in optimizing label models challenge data programmers to achieve comprehensive understanding and well-rounded improvements. Specifically, the main challenge lies in unraveling the intricate relationships among elements (labeling data, LFs, and label models, collectively termed 'elements' for brevity) during labeling iterations. 1). As depicted in Fig. 2, the complexity inherent in many-to-many and directed relationships requires comprehensive relationship analysis since they will interplay in the labeling process. Such relationships and element states are meticulously detailed in Tab. 1. 2). The challenge is further complicated by inconsistent data for-

- *Sisi Li, Tianxiang Wei, Shichao Jia, and Jiawan Zhang are with the College of Intelligence and Computing, Tianjin University, Tianjin 300072, China. Email: sisilee144144@gmail.com; tianxiangwei@tju.edu.cn; jsc_se@tju.edu.cn; jwzhang@tju.edu.cn.*
- *Guanzhong Liu is with Netflix Inc, Los Gatos 95032, United States. E-mail: guanzhongl@netflix.com.*

mats. Since LFs exist as executable codes, attribution for specific label shifts and performance changes becomes intricate. 3). The engagement with large-scale data exacerbates this complexity. Not only does the management and tracing of these relationships become taxing, but evaluations on vast datasets may also be compromised due to potential conflicts among different metrics [8]. For instance, the pursuit of labeling accuracy might diminish coverage across categories.

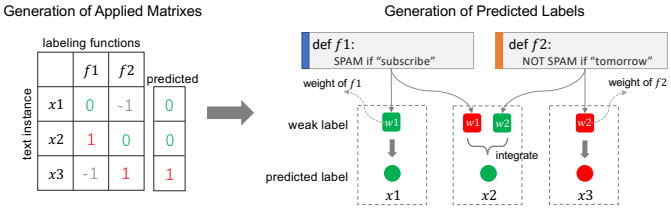


Fig. 2. An example showing how to synthesize labels for text data in data programming while simultaneously demonstrating the involved relationships among elements during the labeling process. For instance, $x2$ is labeled by $f1$ and $f2$ with 1 and 0, finally classified as 0. The Left is an example of an applied matrix. The right presents the generation of predicted labels, where the weighted weak labels are integrated by a label model to assign final probability labels to a subset of data.

Prior work in data programming has focused primarily on two aspects: the intelligent integration of LFs into label models [9], [10], [11] and the facilitation of LF creation with automatic [12], [13] and semi-automatic approaches [14], [15]. While these efforts have made significant strides in promoting data programming, they fall short of capturing the intricate relationships among the elements. Notably, Hoque et al. [5] skillfully integrated image feature distribution and fundamental element relationships to support image labeling in data programming. Similarly, VideoPro [6] fosters the adoption of LF templates and incorporates performance tracing of label models for efficient video data programming. However, these efforts mainly focus on expanding the application scenarios of data programming, without enhancing the technology from an information-transparent perspective. Conducting in-depth relationship analysis during labeling iterations remains challenging due to complex element relationships. Attempts to enhance model interpretability and diagnostic capabilities through visualization techniques [16], [17] have been made, yet these approaches encounter limitations in data programming due to significant workflow and element differences.

To fill this gap, we first identified crucial analysis tasks and design tasks involved in understanding labeling iterations in data programming through expert interviews and related design space exploration. These tasks and design spaces collectively form a framework that enhances data programming techniques from the aspect of interpreting labeling iterations. Furthermore, to concretely demonstrate this framework and validate it in real labeling tasks, we devised and implemented an interactive visual analytics method for multi-class text labeling tasks, EvoVis. This method seamlessly integrates relationship analysis into evolution overviews, presenting contextual and historical information on a single screen. EvoVis supports the process from observation and hypothesis generation to verification and informed decision-making smoothly, in which process data programmers can effortlessly understand and trace the labeling iterations, conduct comprehensive evaluations, and optimize faulty LFs. In summary, our contributions include:

- **A framework for interpreting data programming iterations.** Drawing from expert insights, our generic framework identifies basic analysis tasks, explores comprehensive design spaces, and specifies visualization design tasks for interpreting data programming iterations, which provides a reference for advancing data programming through effective visualizations.
- **A visual analytics method for interpreting multi-class text labeling iterations in data programming.** Based on the framework, we further design and implement EvoVis, an interactive visual analytics system that integrates relationship exploration into labeling history overviews, which effectively facilitates data programmers in text labeling tasks.
- **Efficiency and usability evaluation.** Through multiple case studies and a user study, we demonstrate the utility and effectiveness of EvoVis in improving labeling efficiency using data programming.

2 Related Work

In this section, we review the research related to interactive data labeling, data labeling with data programming, and visual explanation of iterative model building.

2.1 Interactive Data Labeling

Interactive data labeling methods [18] have become essential in the era of AI-driven technologies, serving as a bridge to enhance both the efficiency and quality of data labeling in supervised learning models. Many mixed-initiative approaches were proposed to cope with the widely known challenge of acquiring labeled data. For instance, AILA [19] employed attention-based deep neural networks to facilitate quick and accurate document classification labeling, aiding human labelers in focusing on relevant content. The semantic navigator [20] utilized a visually explainable active learning approach, facilitating human-AI collaboration in zero-shot classification through interactions like asking, explaining, recommending, and responding. In the context of crowdsourcing, Liu et al. [21] introduced an interactive framework to assess and validate uncertain labels, leading to tangible improvements in label quality. In the field of image labeling, Chang et al. [22] devised a spatial layout interface that empowers non-experts to enhance labeled data quality through a strategic spatial organization of images.

While these endeavors offer new insights for various labeling tasks, their direct applicability to labeling tasks in data programming is limited due to fundamental differences in workflow and the forms of elements involved. For instance, while these approaches mainly concentrate on labeling data, LFs, represented by a set of code, serve as pivotal inputs and optimization objects within data programming workflows.

2.2 Data Labeling with Data Programming

To overcome the challenge of obtaining training data, data programming [2] has garnered considerable attention as an

efficient approach for acquiring labeled data through the incorporation of expert knowledge in the form of LFs. Recent related work can be classified into two main categories. The first category aims to design optimized label models that exhibit greater intelligence in weighting LFs and integrating weak labels [9], [10], [11], thereby ensuring the accuracy and practicability of data programming.

The second category focuses on alleviating the high level of programming proficiency required for data programmers [23]. Ratner et al. [24] filtered a subset of the most valuable unlabeled data to inspire users to write LFs based on active learning, resulting in improved performance over a random baseline. Besides highlighting the most informative data, many studies have centered on the avoidance of manual programming. Snuba [12] combined weak supervision and automatic feature learning for more accurate and efficient data labeling. Hancock et al. [13] generated LFs through natural language interpretation labeled by humans. These works significantly relieve the burden on human labelers who adopt data programming in the real world, but still suffer from the readability and accuracy problems of LFs.

Apart from automatic methods, Ruler [14] and TagRuler [15] enable data programmers to efficiently obtain accurately labeled data using predefined concepts for semi-automated LF generation. Both employ highlight visualization to emphasize keywords or concepts in generated LFs. While they ease LF design and show performance changes from previous versions, they lack support for relationship analysis among elements. This makes assessing specific LF impacts on label models challenging and time-consuming, particularly when tracing text data related to certain LFs.

In addition, Visual Concept Programming(VCP) [25] first extended labeling tasks to images with data programming. It generated LFs by integrating visual concepts to label image data, which supports the analysis of some relationships between specific elements involved in VCP. Silimarily, VideoPro [6] not only streamlines the exploration, examination, and application of labeling templates, but it also empowers users to monitor the impact of programming on model performance throughout labeling iterations, which enables efficient programming of video data on a large scale. However, these efforts focus on expanding the application scenarios of data programming, the inadequate relationships and iteration history presented in VCP and VideoPro, continue to pose challenges for data programmers to perform effective evaluation and performance optimization.

In summary, although these efforts have improved labeling efficiency and data quality by combining machine learning(Ml), human-computer interaction, and visualization techniques, as far as our knowledge extends, there is still a research gap in the interpretation of label iterations in the context of data programming. The identification of the optimal direction for improving performance and making astute advancements is elusive. To bridge this research gap, we introduce EvoVis, which presents contextual and historical insights on a unified interface, enabling users to gain a more comprehensive understanding of the iterative labeling process, delve into the underlying causes for unexpected performance outcomes, and arrive at well-informed determinations for further improvements.

2.3 Visual Explanation of Iterative Model Building

Plenty of studies have extensively explored the temporal analysis of the model-optimizing process, which is similar to our work in interpreting the iterations of label models. Hohman et al. [26] attributes model performance changes to iterative data updates. We are inspired by their work and similarly attribute performance fluctuations of label models to updates of LFs. ConfusionFlow [16] displays class confusion matrices with the performance over time to explain and trace the behaviors of classifiers. GANViz [17] combines loss metrics, probability distributions, and activation mapping to offer an insightful exploration and detailed understanding of the training and operational dynamics of Generative Adversarial Networks (GANs). InstanceFlow [27] introduced a dual-view visualization tool that enables users to analyze the iterative learning behaviors of classifiers at the instance level. VISTB [28] allows users to effectively trace prediction evolution, analyze, and compare the importance of data features for tree-boosting models via scalable visualization.

While various visualizations have been introduced to aid the interpretation of the ML model-building process, the distinctive workflow of data programming, characterized by unique LFs, presents inherent challenges in associating performance fluctuations with LF updates. These challenges arise from the complex relationships among elements. For example, in a new round involving modifications to multiple LFs, it becomes challenging to determine which function is responsible for specific changes in label performance or annotation shifts without in-depth relationship analysis. To relieve this challenge, we clarify the main relationships (Tab. 1) among elements by referring to analysis tasks for explaining labeling iterations in data programming, and seamlessly integrate relationship analysis with temporal overviews by multi-view linkage, which effectively facilitates data programmers in understanding labeling iterations.

3 ANALYSIS TASKS AND VISUALIZATION DESIGNS

To discern the primary challenges and analysis tasks in understanding the labeling iterations in data programming, we conducted two rounds of semi-structured interviews with six seasoned ML experts. Our participants encompassed a diverse group, each bringing unique insights and expertise to the study. This group consists of two ML engineers specializing in NLP, both with over 5 years of experience in developing and deploying summarization and QA solutions in the industry, and a data science professor with a distinguished career in ML and big data analysis. The group also includes three Ph.D. researchers specializing in visualization, two of whom have expertise in data programming.

Each interview commenced with an introductory outlining the research goal of our work, followed by a series of pre-defined questions. As the conversations progressed, we tailored our exploratory questions to each expert's unique background and dynamically adjusted based on the discussion to gain deeper insights. The initial 60-minute interviews were subsequently supplemented with 30-minute follow-up sessions to assess the EvoVis prototype to further collect expert feedback on its usability, effectiveness, and potential

TABLE 1
States and relationships among elements in data programming. The diagonal cells depict the main states of each element, while the remaining cells signify the directed relationships between elements. When viewed horizontally, specific elements serve as analysis objects. For instance, A3 represents a focus on a particular instance. It examines which LFs can match this text and provides a candidate label for it. In contrast, B1 concentrates on a specific LF and investigates which instances it can cover. These two relationships are further demonstrated and distinguished by Fig. 2

Data Programming Elements	Labeling Text Data	LFs	Label Model
Labeling Text Data	A1 The data content A2 The distribution of text data	A3 Which LFs can label a specific data A4 Which LFs can contribute to a specific category A5 Which LFs can cause a specific label shift	A6 What data content can interpret a specific label model behavior
LFs	B1 Which data can be labeled by a specific LF B2 What specific weak labels can data be given by a specific LF B3 What categories can be covered by a specific LF B4 Which label shift can be caused by a specific LF	B5 The code content of LFs B6 The label performance B7 The update states B8 The active version extent	B9 How LFs contribute to the building of a label model B10 How to interpret a label model by LF content
Label Model	C1 Which data is labeled as specific categories by a label model C2 What labels can data be assigned by a label model C3 What label shifts are caused by a label model	C4 How to influence the LF updates by performance changes in a label model	C5 The performance of label model

areas for improvement. This prototype stemmed from preliminary analysis tasks and design drafts synthesized from our initial interviews, aimed at refining the essential requirements and expectations of experts. All interviews were audio-recorded for validation and summarization. Based on the expert feedback and as shown in Fig. 3-c, we have identified four core analysis tasks essential for interpreting labeling iterations in data programming.

3.1 Analysis Tasks

T1: Understanding the relationships between main elements. Each element is represented differently and plays a unique role in data programming. For example, a LF, defined as a unit of code, is employed to label multimedia data, including text, images, and video. These elements have distinct states during every round of labeling. Moreover, a complex interplay of many-to-many and directed relationships exists among these elements. A comprehensive understanding of these relationships is imperative for data programmers to analyze how refining LFs can optimize label models, improve the quality of labeled data, and establish a solid foundation for subsequent analysis tasks.

T2: Tracing the history of labeling iterations. Tracing historical labeling in data programming is essential for enhancing the quality of labeled data, as developing label models that meet requirements generally involves multiple iterations of LF optimization. By reviewing past work, data programmers can deepen their understanding of the labeling progress, as well as better capture the characteristics and patterns of labeled data more effectively. Besides, examining historical information can also aid in decision-making regarding version control and determining the optimal stopping point for labeling. Furthermore, exploring past work

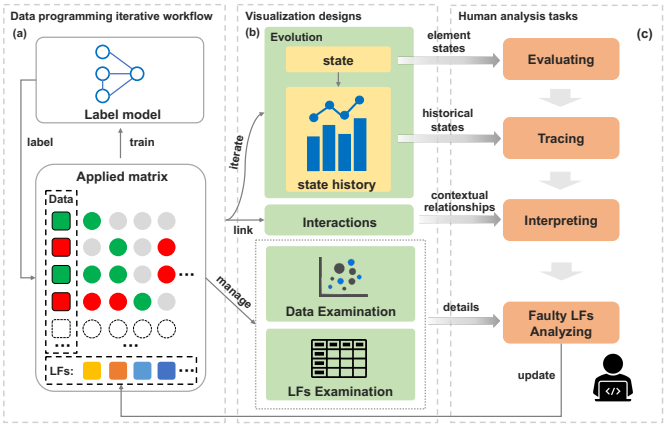


Fig. 3. The framework of EvoVis. a) The simplified labeling process using data programming; b) The visualization designs, which consist of an evolution module, interaction module, and content examination module; c) The human analysis tasks. The evolution module encodes labeling states and histories of elements, which support the evaluation and labeling history tracing tasks. The interaction module is responsible for understanding the relationships between elements, which is essential for explaining labeling behaviors. The examination module manages LFs and data, allowing direct check of their content. This module plays an important role in the verification and decision-making processes in analyzing faulty LFs.

can help prevent redundancy or conflicting work, speeding up the acquisition of satisfactory labeled data.

T3: Comprehensive evaluation of the labeled data. High-quality training data achieve success in essential performance metrics such as accuracy [29], coverage [30], proportional balance [31], and semantic diversity [32]. However, current research on data programming lacks a comprehensive evaluation framework that can consider these

metrics simultaneously. This limitation may lead to missed opportunities to enhance the quality of labeled data. Additionally, persisting with blind iterations may yield little returns. Therefore, there is an urgent demand for a comprehensive method to evaluate labeled data, enabling data programmers to make informed decisions regarding when to terminate iterations and effectively identify areas that require improvement.

T4: Analysis of faulty LFs. LFs with erroneous encoding or inaccurate knowledge expression can adversely affect the performance of label models. Therefore, fault detection holds significance for data programmers to rectify errors and enhance the overall model. Analyzing faulty LFs may reveal specific error types such as missing, duplicate, incorrect, and conflicting errors. Identifying these issues allows for targeted improvements to the LFs. Besides, investigating faulty LFs with historical data may also expose risks associated with improper updates from previous iterations. Despite these insights, there exists a notable gap in research focusing on the analysis of faulty LFs. As a result, debugging remains a time-consuming and formidable task, potentially hindering the widespread adoption of data programming.

3.2 Design Spaces

Given the basic analysis tasks derived from expert interviews, this section explores potential visual design spaces for understanding labeling iterations in data programming. We focus on five dimensions based on visualization and human-computer interaction techniques: attribute, visual coding, interaction, context, and visual representation, providing a comprehensive reference for future investigations.

Attribute Dimension. Exploring the design spaces of attribute dimensions reveals hidden insights in the data programming process, enriching decision support for data programmers. This dimension encompasses performance, dynamic, and configuration attributes: *Performance attributes* include accuracy, recall, and coverage, reflecting the quality of LFs and label models. *Dynamic attributes* trace performance and parameter changes over time in LFs and label models, along with the label shift of labeled data. *Configuration attributes* cover the parameter settings of LFs, such as weight and threshold, which affect their labeling behaviors.

Visual Coding Dimension. Visual coding addresses effective encoding and presentation of data programming elements, encompassing quantitative representation, categorical differentiation, and temporal trend display: *Quantitative coding* can apply static visual codings such as color, shades, and size to highlight differences in magnitude and facilitate comparison; *Category coding* utilizes colors and shapes to clarify different categories; *Temporal coding* employs timelines to reflect the evolution of elements.

Interaction Encoding Dimension. Interaction supports the manipulation of elements, which facilitates the understanding of the complex relationships among elements. This dimension includes navigation, detail examination, and parameter modification: *Navigation* employs various interactive controls, such as drop-down menus, sliders, buttons, etc., to assist data programmers in swiftly locating specific elements, facilitating the pinpointing relevant information and enabling in-depth analysis; *Detail examination* fosters the inspection of detailed information about elements, such as the content and performance of LFs by clicking, hovering, brushing, etc; *Parameter modification* involves adjusting the parameter settings of LFs in real time and observing how these changes affect its performance and labeling results by using slider, checkbox, and text input, etc.

Context Dimension. This dimension aims to provide an overview of the dataset and LFs, detail intra- and inter-element relationships, and iteration history tracing: *Element content overview* provides a snapshot of the entire dataset and LFs with visualization methods such as highlighting; *Intra-element relationships* incorporates interactions between elements, such as conflicting, overlapping, or complementary label behavior between LFs. In addition, it also includes the semantic similarity of labeled data and the distribution across different categories; *Inter-element relationships* refer to the connections between elements, exemplified by the non-diagonal cells in Tab. 1; *Iteration history* displays element changes during iterations, e.g., improvement or degradation of the performance of LFs and tuning of the parameters, changes in the labels of labeled data, etc.

Visualization Expression Dimension. Effective visualization can provide effective representations of complex data, aiding in clear comprehension and insightful analysis. This dimension entails temporal, relationship, and software visualization: *Temporal visualization* utilizes line graphs, flow charts, and heat maps to depict the history of labels, updates, and trends of elements; *Relationship visualization* uses matrix diagrams, graph networks, force-directed graphs, etc. to encode the interconnections between elements, and scatterplots can be applied to express the semantic similarity of the label data and the relationships between the data; *Software visualization* focuses on presenting and organizing LFs by code highlighting and version management approaches.

In summary, this exploration provides a comprehensive view of the design spaces in understanding data programming iterations, highlighting the potential for future research to build upon these insights to develop intuitive and effective tools and techniques.

3.3 Design Tasks

Building on the insights from expert interviews and the exploration of design spaces, we now define specific visualization design tasks (Fig. 3-b) aimed at addressing the identified challenges and directly supporting the improvement of data programming practices. This transition leverages our understanding of data programming's intricacies to inform the development of effective visualization strategies.

D1: States encoding of data programming elements (T1, T2, T3, T4). Clear and intuitive visual representations of element states, such as the coverage and accuracy of LFs within a specific version (T1), are indispensable for evaluating the quality of labeling (T3). Furthermore, presenting specific state information of elements lays the groundwork for tracing labeling histories (T2) and analyzing faulty LFs (T4). The diagonal cells of Tab. 1 illustrate the referable states of elements. For instance, the code content (B5) facilitates the detection of faulty LFs and the interpretation of label behaviors, which can be highlighted with different colors to draw labelers' attention, improve readability, and capture

core concepts of LFs; The accuracy and coverage of LFs (B6) offer valuable insights into the attribution of model performance changes.

D2: Evolution of data programming elements (T2, T3, T4). The overview of evolution not only serves as an effective tool for data programmers to undertake a thorough temporal analysis of labeling history (T2), reducing learning and review expenditures in collaborative development but also enables the detection of degradation relative to previous versions to facilitate the analysis faulty LFs (T4). Moreover, historical performance information is vital for identifying opportunities to enhance data quality (T3) and determining the appropriate moment to cease iterations.

D3: Examination of large-scale multimedia data (T1, T3, T4). The collected data constitutes the core of labeling tasks. Consequently, offering a semantic overview (T1) of the labeled data is instrumental in enabling comprehensive evaluation (T3) and faulty LF analysis (T4). Specifically, assessing the semantic distribution advances the evaluation of the semantic diversity of labeled data. Ideally, the labeling in each category should conform to the overall distribution, reflecting that the generated LFs have encapsulated comprehensive knowledge and classification features. Besides, engaging with raw data facilitates LF debugging and the formulation of recovery strategies for faulty types.

D4: Examination of function-format LFs (T1, T4). As the main input for training label models, LFs serve as a crucial link that connects data and label models. Inspecting their content is essential for elucidating the intricate relationships between elements (T1). Furthermore, managing LFs not only facilitates the identification of potential faulty LFs and error types (T4) but also allows for comparisons between LFs, thereby reducing redundancy and conflicts. Moreover, checking content within the same screen empowers users to conduct reasoning analysis practicably and conveniently.

D5: Multi-view linkage between elements (T1, T4). Multi-view linkage enables relationship exploration across elements (T1), thereby aiding in understanding label behaviors and facilitating the analysis of faulty LFs (T4). As illustrated in Tab. 1, non-diagonal cells represent the directed relationships between elements. For example, for a specific category, which LFs have contributed to it (A4). Similarly, for a specific LF, how many categories has it covered (B3).

4 EvoVis

To facilitate data programmers in understanding the multi-class text labeling iterations, we introduce EvoVis (Fig. 4), a visual analytics system comprising five primary modules. These modules adhere to the design tasks outlined in Section 3.3 and employ appropriate visualization techniques to provide both contextual and historical information on a single screen. A typical example of applying EvoVis to the analysis of faulty LFs is demonstrated in Fig. 5, it encompasses all the analysis tasks defined in Section 3.1

4.1 Evolution Module

We adopt various visualizations to encode the states and historical information of data programming elements (Fig. 4-a) since their data form are completely distinct, offering an intuitive overview of the whole iterative labeling process.

Evolution View of LFs. To present the states and historical changes of LFs, we have designed a two-dimensional visualization that displays the development of LFs (Fig. 4 -a2), where the x-axis represents a selected version extent, and the y-axis indicates the accuracy of LFs (B6). EvoVis employs an ingenuity design of curved lines that connect multiple nodes to visually depict the lifecycle of LFs within a chosen version scope (B8). These nodes can be classified into three types (B7), which are illustrated in Fig. 6. In this representation, pies symbolize code modifications (updated nodes), while unaltered LFs are represented by fixed-size gray dots (hold nodes). The radius of the pies indicates LF coverage, while the distribution of colors within the pies shows the varying ratio of assigned categories (B3).

However, when visualizing multiple LFs within limited space, node overlap may obscure essential patterns and trends. To address this concern, we merge closely positioned nodes, determining their placement based on the average accuracy of the combined set. Specifically, while there are close nodes ($y_2 - y_1 < r$) positioned on $y_1, y_2, ..., y_n$, the labeled subsets can be denoted by $S_1, S_2, ..., S_n$, the new position of merged nodes can be defined as:

$$\overline{y} = \sum_{i=1}^{n} y_i * \frac{\mid S_i \mid}{\sum_{j=1}^{n} \mid S_j \mid}$$

These merged nodes are depicted as black circles, centered with the total number of encapsulated LFs. The size of these merged nodes corresponds to the intersection of their labeled subsets. The integration of merged nodes not only provides an efficient and succinct overview of LF development but also enhances the scalability of LF visualization.

Scaling is employed to switch between detailed information and evolutionary overviews of LFs. Specifically, data programmers are allowed to scrutinize label behaviors and the dynamic evolution of LFs seamlessly by zooming in and out (Fig. 6). Complemented by panning interaction, EvoVis also enables them to conduct targeted investigations into specific regions that potentially harbor patterns. In addition, a highlighting interaction with links is designed to facilitate labelers to be exposed to the lifecycles and states of LFs that pique their interest (Fig. 5-c2). When hovering over a link, the link and the associated nodes will be highlighted, and the merged nodes on that link will split and update to the nodes that primitively belong to the link. This interaction makes it easy for labelers to access the active beginning, end, update state, and labeling performance of LFs by simply hovering over the corresponding links.

Evolution View of Labeled Data Flow. Precisely discerning decision boundaries presents a substantial hurdle in text classification, giving rise to unexpected fluctuations in label assignments across consecutive iterations. Thus, to unveil the outcomes of labeling iterations, we utilize the classical Sankey diagram [33] to transparently visualize shifts in label assignments for text data (C3). As depicted in Fig. 4-a3, the x-axis corresponds to a filtered version range, with each version linked to a distribution of labeled data (C1), visually represented by nodes with distinct colors. The height of these nodes communicates the quantity of labeled data, while label shifts are indicated by links portrayed through seamless color gradients. Additionally,

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3370654

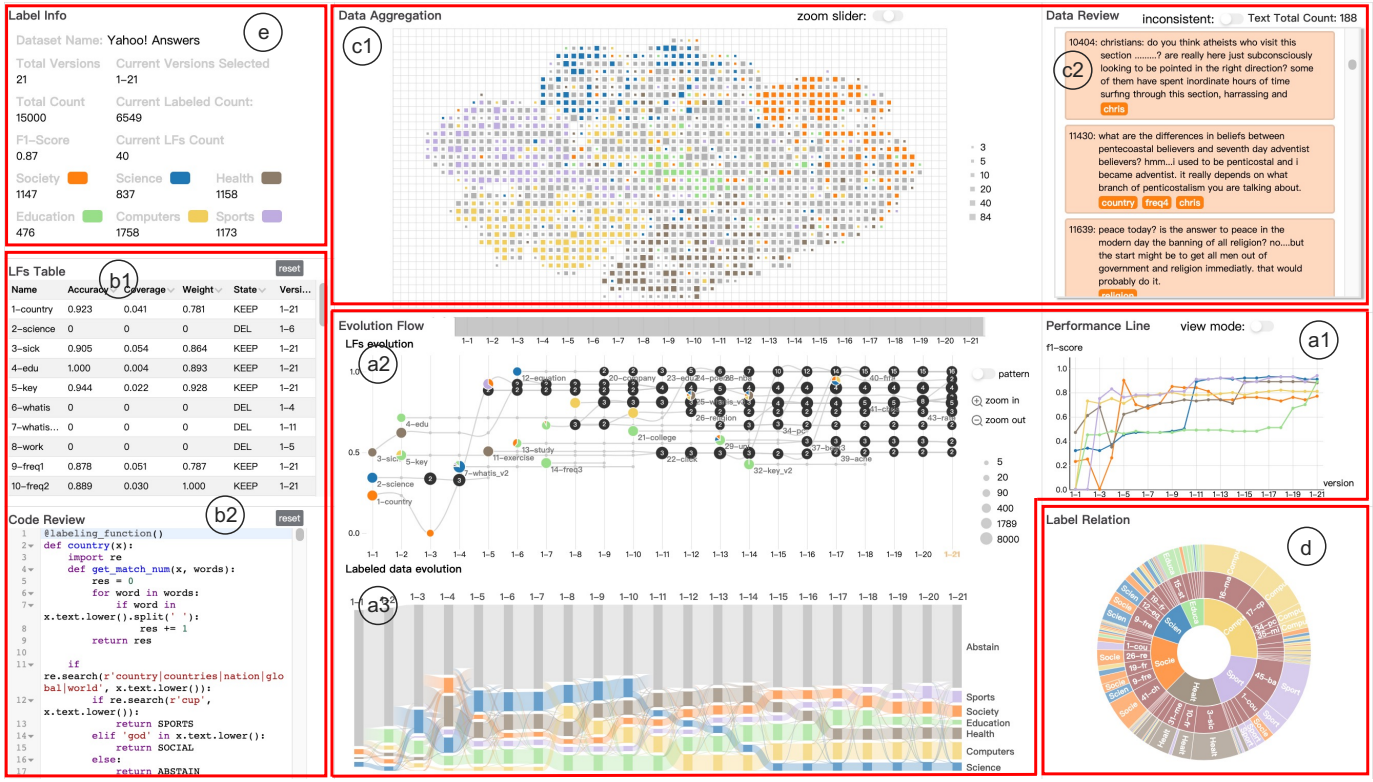


Fig. 4. EvoVis user interface. a) The evolution module illustrates both the labeling states and historical overviews of elements. (a)The evolution model includes three views: (a1) The evolution view of the label model performance, and (a2) The evolution view of LFs, which enables users to explore the distribution and behaviors of LFs by panning and zooming smoothly, and (a3) The evolution view of the labeled data flow. These three views are coordinated by a selected version extent; b) The LF management module comprises two views: (b1) The LFs table view presents the overall states of LFs, and (b2) The code review view displays the highlighted LF content; c) The data management module contains two views: (c1) The data aggregation view locally clusters data with the same categories while maintaining semantics after being projected onto a two-dimensional space, and (c2) The data review view allows users to check the contents of text data; d)The label relation module displays the relationship between data programming elements. e)The label info provides a summary of the current labeling states.

hovering interactions are adopted to emphasize the connections between nodes and links within this representation. In summary, this visualization offers valuable insights to assist labelers in comprehending labeling states, evaluating data quality, and effectively analyzing faulty LFs.

Evolution View of Model Performance. EvoVis employs a simple line graph (Fig. 4-a1) to visually depict the performance trends (C5), with the x-axis denoting the selected versions and the y-axis representing the performance value. Additionally, we have included a slider that allows users to switch between class-wise and overall performance views. By observing whether the model has reached its peak and stabilized, EvoVis can effectively determine the optimal stopping point for labeling tasks. Furthermore, an optimized label model may undergo several dozen or even hundreds of iterations. To streamline the analysis process for any chosen version extent, EvoVis has incorporated a filter bar along with an interactive brush to enable labelers to specify the version range they are interested in, which will be accompanied by updates to other evolutionary views.

4.2 LFs Inspection Module

The LF inspection module (Fig. 4-b) provides an overview of the LF states. This module includes a table view designed for managing and summarizing the attributions of LFs. Moreover, it also provides detailed code review, which allows users to inspect the content of specific LFs.

LFs Table View. As demonstrated in Fig. 4-b1, this table provides a comprehensive overview of the attributes of each LF. Each LF can be uniquely identified by a combination of two components: the inherent function name, which may encounter duplication, especially in collaborative development scenarios, and a numeric ID assigned based on the selected version extent. This unique identifier enables the establishment of relationships between LFs and other elements. The presented attributes of LFs include the last integrated version by the label model and the current version's updated states (B7). A thorough evaluation of LF performance involves assessing their weight, accuracy, and coverage (B6). The weight signifies the contribution of LFs to the label model and is determined by the model (B9). Moreover, users can explore the performance distribution and conduct comparative analyses using the sort buttons adjacent to the headers, facilitating a deeper understanding of LF performance across multiple metrics.

Code Review View. The absence of code examination may potentially impede labelers from accurate explanations regarding the impact of LFs on label models (B10). To address this issue and focus users' attention on the current analysis, EvoVis has incorporated CodeMirror [34], an open-source browser-based code editor, employed to render the

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3370654

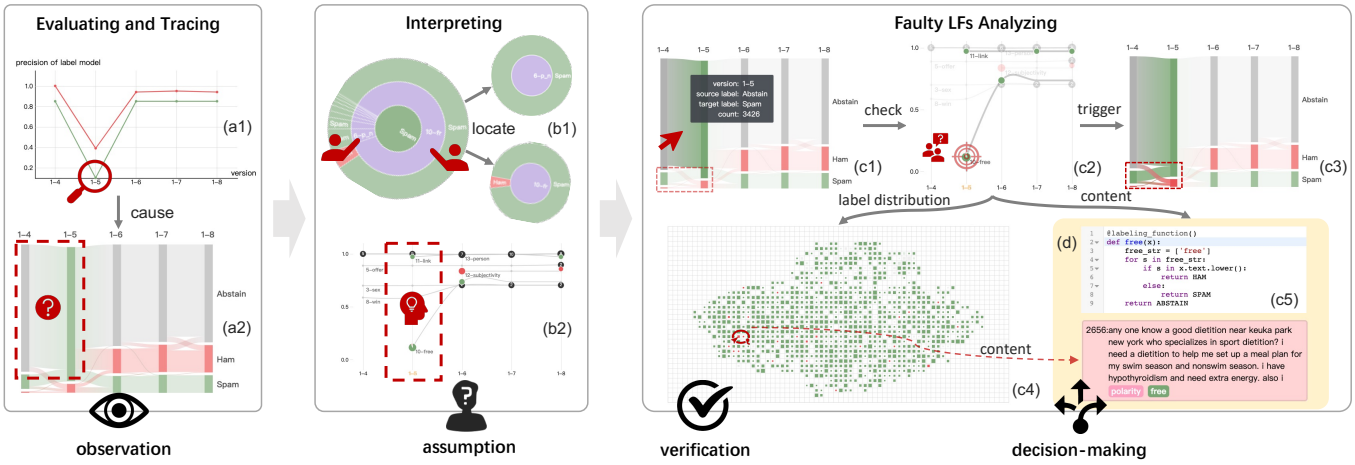


Fig. 5. A typical workflow that includes all analysis tasks using EvoVis. 1) Observation: Identify performance anomalies (a1) and corresponding data flow changes (a2) through the evaluation and tracing of the evolution views; 2) Assumption: Interpret the labeling relationships between different elements (b1) for the faulty version, attribute anomalies in performance and flow changes to function updates (b2), and formulate reasonable hypotheses; 3) Verification: Utilize the abnormal data flow (c1) and multi-view linkage interaction to locate suspected LFs (c2), and verify specific faulty LFs by examining the detailed labeling behavior (c2, c3, c4, c5); and 5) Decision-making: Determine error types and fixes by considering the content of the function and text (d).

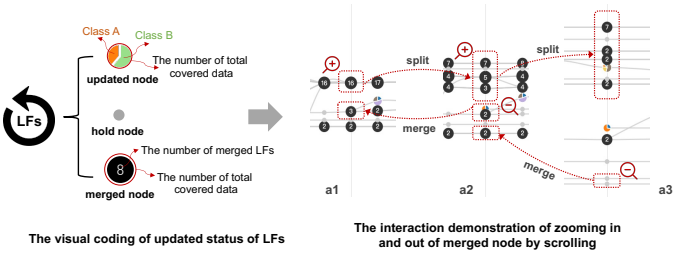


Fig. 6. The illustration of the visual encoding of the states and evolution of LFs. The left shows three types of nodes that represent the updated state of LFs. The right showcases the animation of the splitting and merging of merged nodes triggered by scrolling, an interaction that allows users to conveniently switch between viewing the evolution of LFs at the abstract and detail levels.

codes of LFs (Fig. 4-b2). This powerful tool makes it easy to extract knowledge from these well-formatted and highlighted codes (B5).

4.3 Data Examination Module

The data examination module, as illustrated in Fig. 4-c, consists of two views: the data aggregation view and the data review view. The data aggregation view presents a novel scatterplot that aggregates closely located data with the same categories while preserving global semantics. The data review view assists labelers in checking data content. These two views facilitate direct interactions with the data to be labeled and are crucial for debugging faulty LFs and inspiring future iterations.

Data Aggregation View. Despite various visualization methods that have been attempted to address overplotting and overlay [35] to provide a comprehensive overview of large-scale and multi-class data, maintaining the distribution and density between and within classes without reducing the data size remains a challenge. This issue potentially impacts the accurate assessment of the labeling process. To tackle this challenge, EvoVis incorporates a novel visual abstract (Fig. 4-c1) that aggregates closely located data with consistent categories while preserving global semantics (A2).

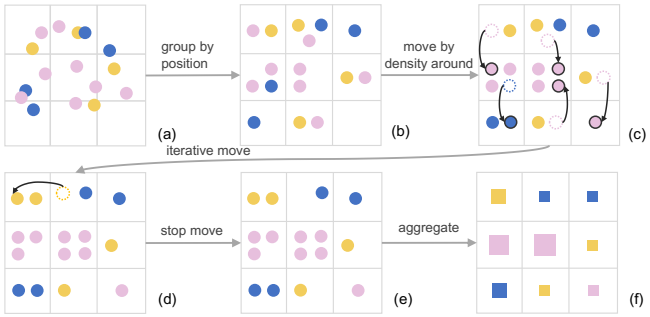


Fig. 7. The iterative aggregating process: a) The space is subdivided into uniform grids; b) Data are assigned to corresponding uniform grids according to their embedding coordinates; c-d) Conflicting categories are iteratively selected and moved in the direction where identical categories or an unoccupied grid are situated; e) The process halts when each grid exclusively accommodates data of a single category; f) The pure data in each grid are grouped and symbolized with squares.

Specifically, we commence by utilizing BERT [36] to procure feature representations for each text. This is followed by the sequential application of PCA [37] and t-SNE [38] for dimensionality reduction and data clustering within a two-dimensional plane. Thereafter, a heuristic algorithm is utilized to locally aggregate data of identical classes. Fig. 7 vividly illustrates this iterative aggregating process. After the aggregation, these data are represented by squares, the size of which correlates with the volume of aggregated data they contain, and their color corresponds to the predetermined label color.

The aggregation algorithm revolves around two key decisions: identifying the data category to be relocated within each grid and determining the direction of its movement. Assuming that a labeling task contains a total of C categories, the mixed data in each grid can be denoted as S_{ij}.

$$S_{ij} = \{D_{ij}^{c} | c = 1, ..., C\}$$

Here, S_{ij} represents data points at position (i, j), and D_{ij}^{c} denotes a subset of S_{ij} with category c. The grids containing mixed categories are systematically traversed to relocate data, resulting in each grid containing data from only one category. To determine the category to be moved first, we calculate the surrounding density of (i, j). Specifically, we utilize $W_{i,j}^{*}$, a fixed-size $r \times r$ window oriented in the direction $*$, to define the density-calculating region. Subsequently, a function $\rho(i, j, c, *)$ is formulated to compute the density of each class $c^{'}$ within the window $W_{i,j}^{*}$.

$$\rho(i, j, c, *) = \frac{\sum_{(p,q) \in W_{ij}^{*}} |D_{pq}^{c}|}{\sum_{(p,q) \in W_{ij}^{*}} \sum_{c=1}^{C} |D_{pq}^{c}|}$$

Theoretically, the category with the highest density is most likely to settle down within a short movement. This approach ensures the preservation of the global distribution by avoiding the displacement of a category that may require significant relocation effort. Therefore, this category is typically chosen as the one to be moved. However, a special case is taken into consideration during this process. When a dominant category c_{dom} exists ($|D_{ij}^{c_{dom}}| > |S_{ij}| * \omega$), where ω is the dominant threshold, this category is excluded from the candidate set of categories to be moved. This strategy prevents the relocation of a substantial amount of data that currently occupies the grid to maintain the pattern of intensive density. The selected category is denoted by $c^{'}$.

Once the category to be moved is determined, we establish a novel scoring mechanism, denoted as $Score$, to find the movement direction. Precisely, we select the direction with the highest score within the grid (i, j) for the relocation of category $c^{'}$. To maintain the formula conciseness, we employ R to represent the tuple of parameters $(i, j, c^{'})$.

$$Score(R, *) = \mu_{\Theta} \cdot \Theta(R, *) + \mu_{\Phi} \cdot \Phi(R, *) + \mu_{d} \cdot \rho(R, *)$$

The indicator function $\Theta(i, j, c^{'},)$ determines whether there is only one category $c^{'}$ in the grid, while the other indicator function $\Phi(i, j, c)$ is used to judge whether the grid is empty. These indicator functions assign a value of 1 if the requirements are satisfied, and 0 otherwise. Additionally, three parameters $\mu_{\Theta}, \mu_{\Phi}, \mu_{d}$ are introduced to control the weights of the three components within $Score$. This enables the adjustment according to specific task requirements. Ultimately, the data of category $c^{'}$ is moved in the direction with the highest-scoring grid. The $Score$ prioritizes moving in directions where the grid either is filled with the identical category or is empty. If neither condition is met, then the movement is directed towards areas with a higher likelihood of finding a concentration of similar categories.

Furthermore, we have included a lasso brush to facilitate the examination of aggregated data. This interaction will promptly trigger the updates of the data review view for checking the labeling results within the semantic space. In addition, a slider is designed to allow users to smoothly switch between three different grid sizes. Smaller grids retain greater details of the original distribution, making them more suitable for nuanced data analysis. Conversely, larger grids result in clearer pattern discernibility between categories, which is more suitable for data overview and exploratory tasks. In summary, this view effectively retains both inter-class and intra-class relationships, facilitating comprehensive evaluation and interaction with a batch of data predicted as the same class.

Data Review View. This view is illustrated in Fig. 4-c2, it allows data programmers to acquire data content (A1). When a text subset is selected, the texts are displayed as individual cards. The background color of each card encodes the labeled result of the label model (C2), and data labeled by different LFs are represented by separate tags (A3, B8), each with a background color signifying the labeled result of LFs (B2). Additionally, by incorporating a toggle bar that facilitates the filtering of inconsistent data, data programmers can selectively inspect the content of conflict data, empowering them to gain profound insights into the label behavior (A6), explore decision boundaries, and effectively debug faulty functions.

4.4 Label Relation Module and View

In EvoVis, a tailored sunburst diagram [39] is employed to provide overviews of some crucial relationships between elements within a specific version, as illustrated in Fig. 8. Categories are differentiated by predefined colors, with each arc encoding the count of labeled items. When focusing on a particular category, the attached outer arc ring signifies a set of LFs contributing to that category (A4), while the outermost ring emphasizes the categories covered by each LF (B3). To expedite the discovery of patterns among elements, the percentage of labeled data within the same context is systematically arranged in a clockwise direction.

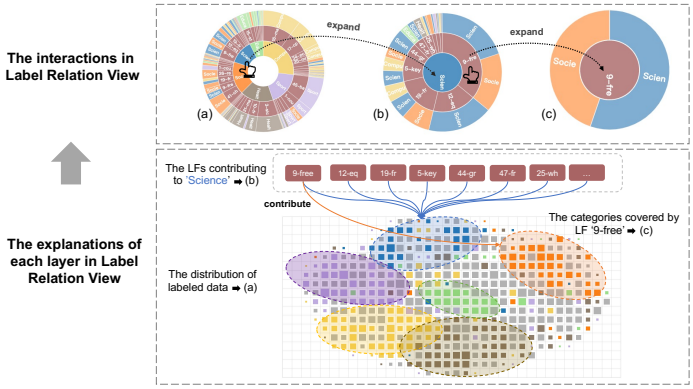


Fig. 8. The demonstration of label relation view: (a) Overview of labeling relationships, with the innermost circle encoding quantity distribution of labeled data by proportion; (b) Exploring the contributions to "Science" by clicking the corresponding arc, related LFs are stretched and positioned around it; (c) Check cover categories of the LF with the highest "Science" contribution by clicking "9-free," revealing its attempt to find a decision boundary between "Society" and "Science." The bottom figure provides further explanations for (a),(b), and (c).

We have incorporated click interaction with animated transitions to facilitate the exploration of specific relationships. When users click on the symbolized category, as illustrated in Fig. 8-b, they can concentrate on the overviews of LFs that have contributed to this category in detail. This

action morphs the clicked ring into a full circle, magnifying attached rings and revealing those previously obscured. This visualization elegantly captures the current distribution of labeled data (C1), pinpoints the LFs responsible for particular categories (A4), exposes the categories covered by each LF (B3), and supports the comparison of the volume of labeled data across categories.

4.5 Multi-view Interactions Module

EvoVis employs a series of multi-view interactions to allow labelers to comprehend and explore the linkages among elements. These interactions can be briefly categorized into those driven by label models, labeled data, and LFs.

Interactions driven by label model. The abnormal performance of the label model often prompts analysis of unexpected data shifts (C3). To identify the causes of labeling exceptions, programmers can first constrain their analysis context by selecting a specific version extent, which triggers an automatic update of the corresponding evolution views. Furthermore, The LFs table view will present a comprehensive list of all the LFs within the selected extent (B8). Moreover, the last version in the version extent is automatically designated as the selected version, which drives the update of the data aggregation view and label relation view based on the selected version.

Interactions driven by labeled data. For the analysis of labeled data within specific categories, data programmers can commence exploration from the evolution view of labeled data flow. When interacting with nodes in this view, the symbolized data is filtered and injected into other views. Specifically, the data aggregation view will highlight the distribution of this data subset in the semantic space (A2), and the text review will populate with the filtered data (A1, B1, C2). Furthermore, LFs contributing to this data subset (A3, C4) will be presented in the LFs table and highlighted in the LFs evolution view. Data programmers are also enabled to analyze label shifts by clicking links, which will update views analogous to the nodes but concentrate on the LFs responsible for the shifts (A5).

Interactions driven by LFs. The LFs evolution and LFs table views offer crucial insights into analyzing the intricate relationships between LFs and other associated elements. Upon selecting a particular function, a cascade of updates will be initiated across multiple views, including the function's content in the code review (B5), the content of applied data in the text review (B1), the highlighting of covered data embedded in the aggregation view (B2), and the highlighting of label shifts influenced by the LF in the evolution view of labeled data (B3).

5 EVALUATION

5.1 Case Studies

In our study, we conducted three case studies to assess the usefulness of EvoVis when it comes to understanding the labeling iterations in data programming with the Yahoo Answers dataset [40]. The origin dataset contains 10 categories, each category contains 140,000 training samples and 6,000 testing samples. To better introduce the designs of EvoVis, we randomly sampled 15,000 text data that can be classified into six distinct topics for concise and reserved 1000 samples for testing purposes. The average length of each text is 102 words, and the topics ranged from the familiar 'Health' to the more complex 'Society'. We employ a color-coded scheme to distinguish these categories, which facilitates the distinction of these categories. Moreover, the LFs are generated by two experienced researchers mentioned in Section 3 and proficient in Python with the seamless support of EvoVis throughout labeling iterations.

5.1.1 Case 1: Evaluation of the Labeled Data Quality

EvoVis can facilitate assessing the accuracy, coverage, balance, and semantic diversity of labeled data in multi-class labeling tasks effectively. Specifically, as depicted in Fig. 4-a1, four categories have achieved an F1 score above 0.9, while 'Computers' and 'Society' may slightly lag behind these four categories. Furthermore, Fig. 4-e presents that in the final version 1-21, a total of 6,549 labeled data points were obtained, accounting for 43.66% of the entire dataset.

However, the distribution of the labeled data is non-uniform. To be more specific, the 'Computer' has a significantly higher quantity of data, approximately four times more than the 'Education'. When this happens, data programmers are not only enable to supplement LFs that cover underrepresented categories based on the current labeling states and expert knowledge but also to expand the feature space by bulk examining unlabeled data surrounding these categories through brush interaction. This approach provides insights to them for extending the new feature space, thereby increasing the data corresponding to the underrepresented categories. Furthermore, navigating from the label relation view to all LFs of an underrepresented class and investing effort in these existing LFs also helps mitigate the issue of class imbalance.

Regarding semantic diversity, we observed in version 1-21 that 'Education' was intertwined with the other five categories, indicating an unclear boundary between 'Education' and other categories. This ambiguity may pose challenges in achieving high accuracy or coverage using solely lexical-based LFs. The distribution of 'Science' is particularly intriguing, with blue blocks interspersed with gray blocks, suggesting the existence of two distinct semantic patterns. Neglecting these unlabeled data could result in incomplete essential training data, potentially hindering ML models from comprehensively learning to distinguish this category accurately from others. Under this circumstance, It would be beneficial to batch-inspect the content of unlabeled data falling within these boundaries to complete the important feature space.

5.1.2 Case 2: Understanding and Tracing the Labeling Process

With the aid of EvoVis, we can conveniently monitor the states and progression of the whole labeling process. By examining the performance shown in Fig. 4-a1, we observed that the categories 'Computers', 'Sports', and 'Society' have already demonstrated commendable F1 scores in early versions 1-5, with the first two reaching a plateau and showing slight improvement throughout the process. The 'Society' category, however, experienced notable fluctuations, with a performance drop followed by a gradual decline until

stabilization. Upon closer examination of the performance view, we found that in versions 1-10, data programmers may have recognized the frustrating labeling performance of 'Science' and 'Education' and then spared no efforts in improving these two categories. After that, the F1 score of 'Education' rapidly increased from below 0.5 to nearly 0.9 in just three consecutive optimizations.

Shifting our focus towards the labeled data flow (Fig. 4-a3) and integrating it with the performance view, we discerned intriguing patterns. Specifically, it came to our attention that data programmers paid little attention to the 'Computers' category before versions 1-8. However, in that specific version, the counts of labeled data surged, and the numbers continued to increase in the next few iterations. Additionally, our analysis revealed a noteworthy observation regarding the 'Health' category. In versions 1-15, A significant amount of data previously labeled as 'Health' was removed from this category and the number remained constant until the end. This observation suggests that data programmers made substantial corrections in that version to improve the performance of 'Health' labeling, and the subsequent labeling process did not uncover any new patterns for this category.

Upon careful examination of the LFs evolution presented in Fig. 4-a2, a remarkable observation emerged. Before versions 1-10, almost half of the LFs failed to attain an accuracy of 0.6, resulting in an unsatisfactory performance for the label model. However, through multiple iterations, a substantial majority of the LFs demonstrated significant improvement. In the latest version, 30 LFs achieved an accuracy of 0.8 or higher, constituting 75% of all LFs. These findings emphasize the value of iterative LF enhancement as a means to improve the quality of labeled data.

Fig. 4-d provides a concise summary of the intricate relationships among the elements in the final version. When selecting the circle associated with the most labeled category, 'Computers', all LFs that contributed to generating labeled data for this category were arranged clockwise based on their quantities. From this view, it becomes evident that F16 and F17 were the primary LFs responsibly contributing to labeling this category, upon closely examining these two LFs, we ascertained that they were deliberately crafted to cater to the labeling needs of 'Computers'.

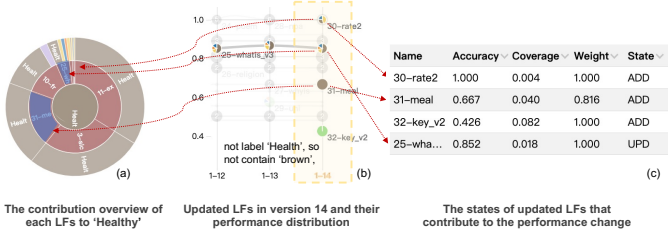


Fig. 9. The potential LFs that lead to 'Health' degradation. (a) The distribution of contributed LFs regarding 'Health' leads to hypotheses formulation based on the amount of contribution; (b) The performance distribution of updated LFs helps to locate possible faulty LFs; (c) The detailed attribution and performance of updated LFs provide additional information on LFs.

5.1.3 Case 3: Analysis of the Faulty LFs

The aforementioned case has demonstrated the considerable impact of LF quality on the label models' performance. In this context, EvoVis proves to be a valuable tool as it expedites the identification and debugging of faulty LFs. To better illustrate this point, we will delve deeper into the performance degradation of the 'Health' category in versions 1-14 (Fig. 4-a1).

To detect potential faulty LFs, we first examine the evolution of LFs (Fig. 9-b), which reveals that during versions 1-14, four LFs underwent updates, with three of them associated with the 'Health' category, indicated by a brown-colored segment in the pies. Among these, F30 occupied the top position, boasting an accuracy of nearly 1. Given its size, we can deduce that F30 is an expected labeling function, owing to its high accuracy and low coverage, requiring no further adjustments. F25 and F31 lie below F30, prompting us to surmise that both of them may have contributed to the decline in 'Health's performance. Fig. 9-a shows that F31 encompassed a larger data volume compared to F25, indicating a more drastic update. Moreover, considering F31's accuracy merely reaching 0.65, our speculation leans towards F31 exerting a potentially more pronounced impact.

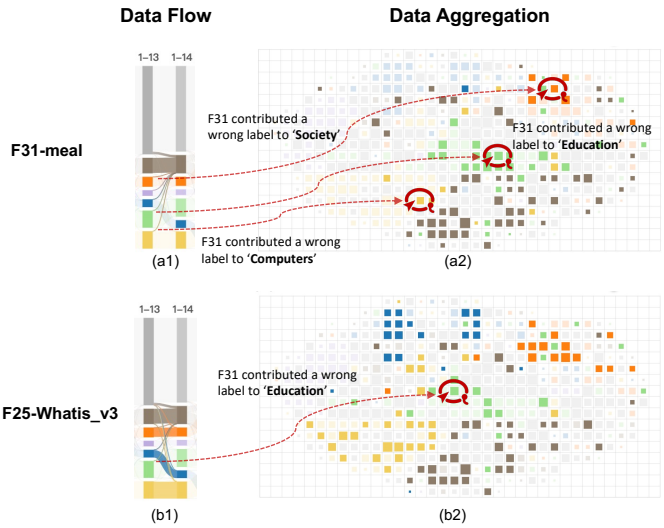


Fig. 10. The impact on labeled data by F31 and F25. The left shows the flow of label shifts driven by F31 and F25; The right presents the distribution of data labeled by F31 and F25 in semantic space.

Upon examining data flows driven by F31 and F25 in Fig. 10, we find that F31 is a pure function that contributes to the label shifts from the other five categories to 'Health', while F25 is a mixed function that leads to a mess of label shifts. Subsequently, we highlighted data labeled by target LFs in the semantic distribution view to explore the distribution of these data. However, F31 contributes quantities of incorrect labels to many other categories, especially 'Education', 'Society', and 'Computers', as evidenced by many non-brown colored blocks. This suggests that F31 enhances the coverage of 'Health' (Fig. 10-a1) at the expense of accuracy (Fig. 10-a2). As for F25, although its distribution (Fig. 10-b2) aligns with our expectations, a few green blocks indicate that F25 may move data from 'Education' to 'Health' by mistake.

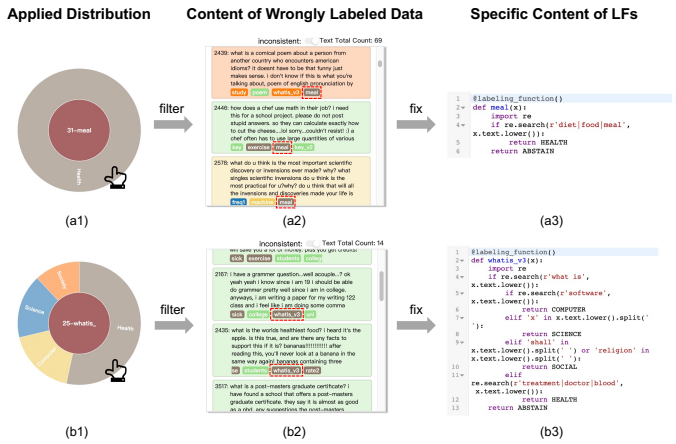


Fig. 11. Comparative analysis between F31 and F24 for locating faulty LF. Through interaction with the label relation view, instances where candidate LFs were incorrectly labeled as "Health" were selected. The predicted labels for these instances conflicted with the candidate labels provided by potential faulty LFs. Combining the specific text content of conflicted data with the LF content revealed that F31 led to more conflicts, involved a broader range of mislabeled categories, and exhibited less rational rule design. This analysis process efficiently assists data programmers in locating and optimizing faulty LFs.

Based on the above analysis, we conclude that the decrease in labeling accuracy for the 'Health' in versions 1-14 can be primarily attributed to F31, with minor issues stemming from F25. To further verify our assumption and gain insights into fixing faulty LFs, we focused on the analysis of mislabeled data content (Fig. 11) by interacting with the relation view to filter out conflicted data mislabeled as 'Healthy' by target LFs. During this process, we identified data about food and dietary knowledge mislabeled as 'Education' by F31, and some texts related to religious dietary habits mislabeled as 'Society'. Combining this observation with the code content of F31 (Fig. 11-a3), we recommend adding more rules to F31 to identify specific food-related terms, such as "nutrition", "food group" and "calories".

5.2 User Study

To perform an objective evaluation of the effectiveness and usefulness of EvoVis, our study incorporated both quantitative and qualitative analyses in practical labeling tasks. Specifically, we continue to employ the processed Yahoo Answer dataset, which is elaborated in Section 5.1. Given the typical workflow of data programming entails the development of Python functions, we invited 40 participants who possess basic programming proficiencies, comprising 8 individuals with PhDs (P1-P8), 26 graduate students (P9-P34), and 6 undergraduates (P35-P40), 8 of whom are female. Half of the participants have been exposed to labeling tasks before, with 23 of them boasting over a year of Python programming experience, and 7 participants even more than three years. Despite their unfamiliarity with data programming, our observations revealed that participants with relevant backgrounds could adeptly embrace this technology within a brief time. These participants were evenly divided into control and experimental groups.

The ultimate goal for participants was to maximize the F1 score of label models through iterative refinement of label models while considering the overall coverage. Thus, for the control group, beyond the standard performance metrics encompassed in Snorkel [23], such as coverage, overlap, and conflict rate for each LF, the F1 score and coverage for each category were also furnished to facilitate participants during the labeling process. Regarding the experimental group, after acquainting themselves with EvoVis, they were granted to seek EvoVis for assistance. The total experiment spanned approximately one hour, with each participant receiving a minimum hourly remuneration of $7.25 as compensation.

To mitigate the significant impact of coding proficiency on the application of data programming techniques(one possesses domain knowledge but struggles with the labeling task due to an inability to formulate precise LFs is an undesirable scenario), we offered 64 candidate labeling functions of varying quality for participants to choose from. Furthermore, we also devised a dedicated operating platform that allowed the customization of LFs. Notably, we defined each iteration of model training as a distinct version. Updates of a labeling function are identified through comparison with its counterpart from the last version.

Before the formal experiment, all participants were afforded a brief 5-minute introduction to the core principles and workflow of data programming. Subsequently, they were immersed in the requisite preparatory training. Specifically, we demonstrated the process of training a label model using the operating platform. Thereafter, for the control group, the standard performance metrics were introduced, and two concrete analysis tasks involving evaluation and potential faulty LF detection (T3, T4) were conducted to facilitate the training process for the impending experiment. The training process for the control group lasted about 15 minutes. In the case of the experimental group, an initial 20-minute workshop was orchestrated to introduce the design principles and functionalities of EvoVis, followed by the demonstration of the analysis tasks (T1-T4) with EvoVis, aimed at enhancing their acquaintance with EvoVis. After the training, a 10-minute pre-experiment of binary classification was conducted, enabling them to familiarize themselves with the labeling process with their respective tools. Throughout this stage, all posed questions were duly addressed. Ultimately, they were inducted into a formal labeling task. Specifically, participants were required to complete a 30-minute labeling experiment, during which any raised inquiries were recorded but left unanswered. At the end of the study, we administered a post-experiment questionnaire to collect feedback on EvoVis. Additionally, we conducted 10-minute one-on-one interviews with each participant to summarize their evaluations and suggestions.

5.2.1 *Quantitative Evaluation*

The results of the user study are shown in Fig. 12 and 13. In Fig. 12, a comparison of F1 scores is made between the control group and the experimental group across six categories. Notably, across all six categories, the experimental group exhibits higher median F1 scores. This observation strongly implies that EvoVis yields a substantial positive impact on labeling tasks in data programming.

In the categories of 'Science,' 'Health,' and 'Computers,' the experimental group demonstrates both a higher median performance and greater consistency, marked by a

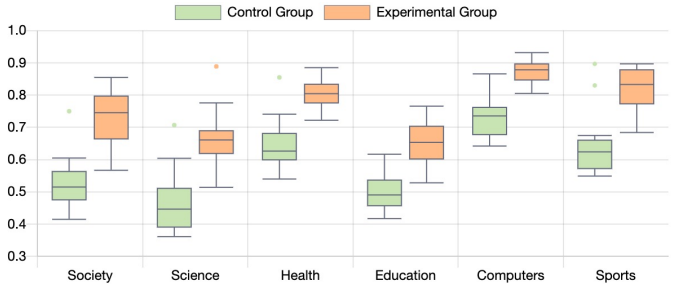


Fig. 12. The distribution of F1 scores across various categories for experimental and control groups.

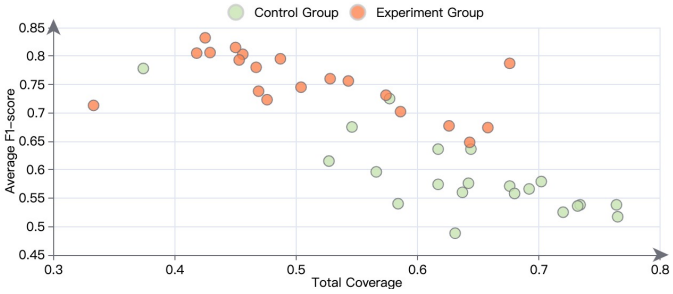


Fig. 13. The distribution of F1 scores and total coverage of the experimental and control groups.

reduced interquartile range (IQR), signifying more robust and reliable results with EvoVis. Particularly in 'Health' and 'Computers,' where labeling is relatively straightforward, both groups achieve elevated F1 scores. EvoVis further distinguishes itself by providing insightful visualizations, enabling labelers to recognize and focus on relevant features, contributing to greater accuracy in these domains.

Within the categories of 'Society,' 'Science,' and 'Sports,' though the experimental group's IQR expanded, it is salient that its lower quartile remains positioned above the median of the control group. This observation underscores that, despite the broader distribution of F1 scores in these domains, the incorporation of EvoVis bestows an inherent advantage to the data labeling process. The increase in IQR may be ascribed to the unique complexity characterizing the data within these domains. Assisted by EvoVis, the experimental group was enabled to explore more intricate instances (e.g., those lying at the boundaries between categories), thus resulting in a broader IQR of F1 scores. Such heightened variability is indicative of the experimental group's proficiency in managing diverse and multifaceted data points, highlighting the tangible value of EvoVis in this context.

Fig. 13 illustrates the distributions of total coverage(the percentage of text labeled within the corpus.) and average F1 score for both the experimental and control groups. The graph reveals a negative correlation between total coverage and average F1 score for both groups, with the experimental group consistently registering higher average F1 scores, irrespective of total coverage. This discrepancy may stem from the control group's struggles in analyzing statistical performance and decision-making, hindered by the absence of intuitive temporal information, interpretation of labeling behaviors, and performance evaluation. Conversely, those utilizing EvoVis were guided by evolutionary information, focusing more on refining existing LFs for superior performance while considering coverage. This approach enabled them to avoid the pitfalls of the control group, who relied on general LFs that matched large data sets but failed to differentiate between categories effectively. Notably, within the coverage range of 0.5 to 0.65, the experimental group's data is markedly distributed above the control group's at the same coverage, further affirming the practicality and efficacy of EvoVis.

An outlier achieved the highest F1 score within the control group and a coverage of 0.4 emerged. Post-experiment insights attribute this result to the participant's relentless focus on the careful optimization of each LF before introducing new ones. He prioritized the F1 performance for each incorporated LF, with little regard to overlap and coverage. Yet, despite his meticulous efforts, several participants in the experimental group attained comparable or even superior F1 scores at higher coverage levels. This emphasizes EvoVis's effectiveness in not only improving data quality but also in achieving remarkable results at broader coverage.

5.2.2 *Qualitative Evaluation*

Apart from quantitative evaluation, we also explore the qualitative examination of EvoVis, focusing on its usability and effectiveness. The results from post-experiment questionnaires, the feedback received from interviews with participants, and the observations during experiments will be discussed to provide a comprehensive evaluation of EvoVis.

As delineated in Fig. 14, the questionnaire outcomes demonstrate that EvoVis fosters user comprehension in labeling tasks, with a mean score of 4.65, and 70% of participants awarding the maximum score (Fig. 14-a). For example, participant P12 shared his experience: "I was a little confused by the practice of data programming at the beginning, but the real-world application and analysis task demonstration used by EvoVis swiftly allowed me to immerse myself in the role of a proficient data programming expert." Besides, as Fig. 14-b illustrates, all participants concur that EvoVis facilitated a more efficient labeling process. Participant P2, for instance, said "EvoVis makes it easier and smoother to complete the labeling task". Another participant, P22, offered insight by stating, "With EvoVis, I always knew how to move further with a clear objective in my experiment". These comments attest to the efficacy of EvoVis in data programming. Conversely, Fig. 14-c shows that a subset of users found EvoVis challenging,

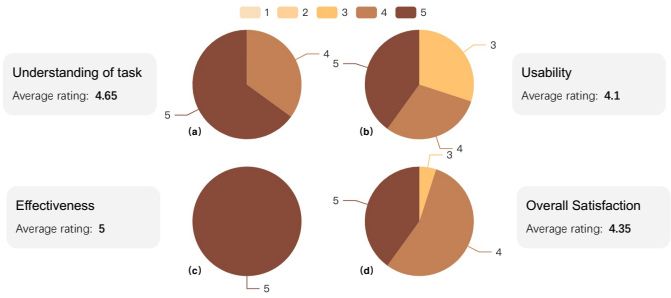


Fig. 14. The distribution of participants grading in terms of the understanding of labeling tasks, the usability, effectiveness, and overall satisfaction of EvoVis.

rating its usability at 3. These users pointed out difficulties stemming from the presence of multiple views and intricate element relationships, necessitating a certain investment in learning. "Although the pre-training and pre-experimental are really helpful for initial familiarization, I still can not fully understand the design principles and how to use this system to optimize the label model. Until the mid-experiment, I get along with EvoVis to engage with my task", P17 commented. Despite these challenges, the overall satisfaction score of 4.35 (Fig. 14-d) underscores the considerable contentment with EvoVis for participants.

During the experiment, some interesting observations also deserve to be discussed. When analyzing the results of experiments, we found that in the experimental group, even a Ph.D. student with more than 3 years of programming experience and experience in data labeling may generate labeled data of lower quality than a sophomore, undergraduate student with less than 1 year of programming experience and no experience in data labeling. This suggests that with the assistance of EvoVis, the requirement of the labelers' ability is substantially weakened. Besides, a notable subset of participants within the control group adopted a traditional approach, resorting to pen and paper for documenting changes in performance and function content. Interestingly, out of the total 20 participants in experiment groups, about 7 participants undertook the endeavor of modifying function content. Remarkably, one participant, characterized by an engagement with deeper usage of EvoVis and then achieved a substantial enhancement in performance. This participant highly praised the effective process from observation, assumption, and verification to informed decision-making by taking advantage of EvoVis's multiple views to update specific LFs with confident orientation.

When shifting focus to labeling strategies, three user preferences became apparent. Some users prefer the incremental training paradigm, involving a systematic step-by-step process. In contrast, another faction favored a strategy involving the incorporation of all LFs before the subsequent identification and removal of faulty LFs. Some participants chose to begin extensive modifications to function content after selecting a few initial functions. We found that the second strategy resulted in relatively higher efficiency, as it can avoid spending a lot of time analyzing low-quality LFs by taking full advantage of performance attribution and fault LF fixture provided by EvoVis.

6 DISCUSSION

6.1 Implications

Our work has several important implications. First, we address a critical research gap. Given the growing importance of data-centric AI in modern AI development, effectively utilizing data programming for acquiring training data is essential. Enhancing data programming through visualization, particularly focusing on interpreting its labeling iterations, is beneficial for its broader adoption. While recent works [6], [25] have covered aspects related to tracing labeling history and exploring element relationships, their primary emphasis remains on enhancing the efficiency and quality of image and video labeling using data programming. In contrast, our research serves as a crucial complement, focusing on gaining a comprehensive understanding of labeling states, facilitating thorough assessments, and enabling the identification and rectification of faulty LFs.

Second, we offer a concrete visual analytics method equipped with instructional and validation capabilities for multi-class text labeling tasks. The results from three case studies and a user study illustrate and prove how data programmers can effectively perform labeling tasks and enhance data quality through a better understanding of the labeling iterations.

Third, the visualization methods in EvoVis provide valuable reference points. For instance, in large-scale, multi-class data labeling tasks, we design a visual abstract that locally aggregates multi-class points. This approach effectively preserves global semantics while presenting inter-class relationships and comparative information within data points, deserving further expansion and validation. Furthermore, this visual design, along with the LF evolution view, offers great scalability to our work.

6.2 Limitations and Future Work

Assumptions about the testing dataset: Certain evaluation metrics in our study are based on the existence of a testing dataset. However, in real-world labeling scenarios, there is often limited data available in the early stage, which may result in biases and imbalances. In such cases, we cannot guarantee the performance of the label model necessarily represents the quality of the labeled data and LFs.

Insufficient intelligence in the exploration process for non-expert users: EvoVis is primarily designed for data analysts who are familiar with the workflow of data programming and possess the capability to efficiently associate elements. However, non-expert users may find the system overwhelming due to the large amount of information presented. Specifically, when conducting faulty LF analysis, non-expert users may struggle to identify them, which can impede their ability to improve these LFs.

In the future, we plan to improve and generalize three aspects: (1) Applying EvoVis to other ML tasks, we aim to broaden its usage across various application scenarios as data programming evolves to support tasks like image classification and video data programming. We believe our method demonstrates strong generalizability, allowing for its labeling interpretation extension to other tasks with minimal modification; (2) Incorporating advanced analysis capabilities for error analysis, we believe automatic attribution and inference could further enhance the system's usability. For instance, evaluating and ranking LFs based on their overall performance within a selected version automatically by focusing on criteria like coverage and conflict rates within specific categories; (3) Inspecting the testing dataset to help with the early-stage labeling, such as integrating the semantics and category distribution of the testing dataset into EvoVis without exposing the specific content of the dataset, to improve the quality of early-stage labeled data.

7 CONCLUSION

In this paper, we propose a generic framework for enhancing data programming techniques from the perspective of

understanding the iterative labeling process. Based on this framework, we designed and implemented EvoVis, a visual analytics method specifically designed for multi-class text labeling tasks. EvoVis consists of five modules to simultaneously present contextual and historical information, which collectively provide a user-friendly environment to assist data programmers in performing labeling tasks by facilitating the comprehension, tracing, evaluation, and faulty LF analysis throughout the labeling iterations. We evaluate the utility and effectiveness of EvoVis from both quantitative and qualitative perspectives, demonstrating its capability to efficiently improve the quality of labeled data.

REFERENCES

[1] Z. Wang, H. You, J. Chen, Y. Zhang, X. Dong, and W. Zhang, "Prioritizing test inputs for deep neural networks via mutation analysis," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 397–409.

[2] A. J. Ratner, C. M. De Sa, S. Wu, D. Selsam, and C. Ré, "Data programming: Creating large training sets, quickly," *Advances in neural information processing systems*, vol. 29, 2016.

[3] J. Zhang, C.-Y. Hsieh, Y. Yu, C. Zhang, and A. Ratner, "A survey on programmatic weak supervision," *arXiv preprint arXiv:2202.05433*, 2022.

[4] Y. Wang, S. Sohn, S. Liu, F. Shen, L. Wang, E. J. Atkinson, S. Amin, and H. Liu, "A clinical text classification paradigm using weak supervision and deep representation," *BMC medical informatics and decision making*, vol. 19, pp. 1–13, 2019.

[5] J. Huang, A. Mishra, B. C. Kwon, and C. Bryan, "Conceptexplainer: Interactive explanation for deep neural networks from a concept perspective," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 831–841, 2022.

[6] J. He, X. Wang, K. K. Wong, X. Huang, C. Chen, Z. Chen, F. Wang, M. Zhu, and H. Qu, "Videopro: A visual analytics approach for interactive video programming," *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[7] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[8] H. Chen, J. Chen, and J. Ding, "Data evaluation and enhancement for quality improvement of machine learning," *IEEE Transactions on Reliability*, vol. 70, no. 2, pp. 831–847, 2021.

[9] S. H. Bach, B. He, A. Ratner, and C. Ré, "Learning the structure of generative models without labeled data," in *International Conference on Machine Learning*. PMLR, 2017, pp. 273–282.

[10] A. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey, and C. Ré, "Training complex models with multi-task weak supervision," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 4763–4771.

[11] P. Varma, F. Sala, A. He, A. Ratner, and C. Ré, "Learning dependency structures for weak supervision models," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6418–6427.

[12] P. Varma and C. Ré, "Snuba: Automating weak supervision to label training data," in *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, vol. 12, no. 3. NIH Public Access, 2018, p. 223.

[13] B. Hancock, M. Bringmann, P. Varma, P. Liang, S. Wang, and C. Ré, "Training classifiers with natural language explanations," in *Proceedings of the conference. Association for Computational Linguistics. Meeting*, vol. 2018. NIH Public Access, 2018, p. 1884.

[14] S. Evensen, C. Ge, and C. Demiralp, "Ruler: Data programming by demonstration for document labeling," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 1996–2005.

[15] D. Choi, S. Evensen, Ç. Demiralp, and E. Hruschka, "Tagruler: Interactive tool for span-level data programming by demonstration," in *Companion Proceedings of the Web Conference 2021*, 2021, pp. 673–677.

[16] A. Hinterreiter, P. Ruch, H. Stitz, M. Ennemoser, J. Bernard, H. Strobelt, and M. Streit, "Confusionflow: A model-agnostic visualization for temporal analysis of classifier confusion," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 2, pp. 1222–1236, 2020.

[17] J. Wang, L. Gou, H. Yang, and H.-W. Shen, "Ganviz: A visual analytics approach to understand the adversarial game," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 6, pp. 1905–1917, 2018.

[18] B. Grimmeisen, M. Chegini, and A. Theissler, "Visgil: machine learning-based visual guidance for interactive labeling," *The Visual Computer*, pp. 1–23, 2022.

[19] M. Choi, C. Park, S. Yang, Y. Kim, J. Choo, and S. R. Hong, "Aila: Attentive interactive labeling assistant for document classification through attention-based deep neural networks," in *Proceedings of the 2019 CHI conference on human factors in computing systems*, 2019, pp. 1–12.

[20] S. Jia, Z. Li, N. Chen, and J. Zhang, "Towards visual explainable active learning for zero-shot classification," *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 1, pp. 791–801, 2021.

[21] S. Liu, C. Chen, Y. Lu, F. Ouyang, and B. Wang, "An interactive method to improve crowdsourced annotations," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 1, pp. 235–245, 2018.

[22] C.-M. Chang, C.-H. Lee, and T. Igarashi, "Spatial labeling: leveraging spatial layout for improving label quality in non-expert image annotation," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–12.

[23] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré, "Snorkel: Rapid training data creation with weak supervision," in *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, vol. 11, no. 3. NIH Public Access, 2017, p. 269.

[24] B. Cohen-Wang, S. Mussmann, A. Ratner, and C. Ré, "Interactive programmatic labeling for weak supervision," in *Proc. KDD DCCL Workshop*, vol. 120, 2019.

[25] M. N. Hoque, W. He, A. K. Shekar, L. Gou, and L. Ren, "Visual concept programming: A visual analytics approach to injecting human intelligence at scale," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 74–83, 2022.

[26] F. Hohman, K. Wongsuphasawat, M. B. Kery, and K. Patel, "Understanding and visualizing data iteration in machine learning," in *Proceedings of the 2020 CHI conference on human factors in computing systems*, 2020, pp. 1–13.

[27] M. Pühringer, A. Hinterreiter, and M. Streit, "Instanceflow: Visualizing the evolution of classifier confusion at the instance level," in *2020 IEEE visualization conference (VIS)*. IEEE, 2020, pp. 291–295.

[28] J. Wang, W. Zhang, L. Wang, and H. Yang, "Investigating the evolution of tree boosting models with visual analytics," in *2021 IEEE 14th Pacific Visualization Symposium (PacificVis)*. IEEE, 2021, pp. 186–195.

[29] J. Nivre, M.-C. De Marneffe, F. Ginter, Y. Goldberg, J. Hajic, C. D. Manning, R. McDonald, S. Petrov, S. Pyysalo, N. Silveira *et al.*, "Universal dependencies v1: A multilingual treebank collection," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 2016, pp. 1659–1666.

[30] J. D. Kelleher, B. Mac Namee, and A. D'Arcy, "Fundamentals of machine learning for predictive data analytics: algorithms, *Worked examples, and case studies*, 2015.

[31] C. Huang, Y. Li, C. C. Loy, and X. Tang, "Learning deep representation for imbalanced classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5375–5384.

[32] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Conference on fairness, accountability and transparency*. PMLR, 2018, pp. 77–91.

[33] M. Schmidt, "The sankey diagram in energy and material flow management: part ii: methodology and current applications," *Journal of industrial ecology*, vol. 12, no. 2, pp. 173–185, 2008.

[34] M. Haverbeke, "CodeMirror: A versatile text editor implemented in JavaScript for the browser," https://codemirror.net/, n.d., accessed: 2023-01-28.

[35] A. Sarikaya and M. Gleicher, "Scatterplots: Tasks, data, and designs," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 402–412, 2017.

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational

Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[37] I. Jolliffe, *Principal Component Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1094–1096. [Online]. Available: https://doi.org/10.1007/978-3-642-04898-2_455

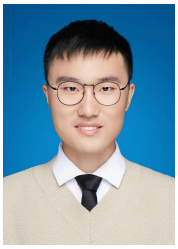
[38] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html

[39] J. Yang, M. O. Ward, and E. A. Rundensteiner, "Interring: An interactive tool for visually navigating and manipulating hierarchical structures," in *IEEE Symposium on Information Visualization, 2002. INFOVIS 2002.* IEEE, 2002, pp. 77–84.

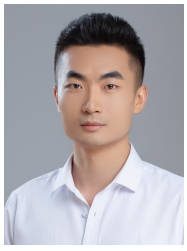
[40] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 649–657.

Sisi Li received her BS degree from software engineering at Tianjin University in 2021. She now is pursuing her MS degree in Computer Science at Tianjin University. She focuses on applying various visualizations to interpret machine learning models.

Guanzhong Liu received his MS degree from Language Technologies Institute at Carnegie Mellon University in 2023. His main research interests include natural language processing, information retrieval, and visualization. Currently, he is working at Netflix, Inc.



Tianxiang Wei received his B.Sc. degree from Jinling Institute of Technology in 2017, and the M.Sc. degree in computer science at Zhejiang Gongshang University in 2020. He is pursuing a Ph.D. degree from Tianjin University. His research interests mainly include scientific visualization.



Shichao Jia received his Ph.D. degree from Tianjin University. His research lies at the intersection of information visualization and machine learning, with a focus on leveraging visual analytics to understand and steer deep learning models interactively and applying machine learning advances to data visualization.

Jiawan Zhang is a professor at the College of Intelligence and Computing, Tianjin University. Leading the Visual Computing Lab and the Institute of Data Science, he also directs the Tianjin Key Engineering Center for Cultural Heritage Conservation and Promotion. His research focuses on computer graphics and visual analysis. He has served as chair or co-chair for several IEEE and other international conferences and is actively involved in the academic community as a reviewer and committee member.