Towards Explainable Table Interpretation Using Multi-view Explanations

Yunjun Gao[1], Pengfei Wang[1*], Xiaocan Zeng[1*], Lu Chen[1✉], Yuren Mao[1], Ziheng Wei[2], Miao Li[2]
[1]*Zhejiang University*, [2]*Huawei Cloud Computing Technologies Co., Ltd.*
{gaoyj, wangpf, zengxc, luchen, yuren.mao}@zju.edu.cn, {ziheng.wei, limiao12}@huawei.com

***Abstract*— Table interpretation (TI), which aims to predict the column types and relations of tables, plays an essential role in necessary decision-making actions for data management systems. Typically, TI is followed by a manual verification, where experts manually verify the correctness of TI's predictions. Manual verification is able to ensure the quality of decision-making actions but labor-intensive. To reduce the labour costs, providing explanations for TI's predictions is necessary as these explanations can help them do faster and more accurate verification. However, existing TI approaches overlook the manual verification process and lack explainability[1]. To fill this gap, this paper explores the challenging explainable table interpretation problem, which aims to provide faithful explanations and meanwhile achieve high prediction performance. Furthermore, we propose ExplainTI framework. ExplainTI consists of two phases: (i) tables are converted to sequences and lightweight column graphs; and (ii) a pre-trained transformer encoder is fine-tuned to provide multi-view explanations and aggregate contextual information. Extensive experiments on both real Web tables and database tables confirm that ExplainTI outperforms competitive baselines. Moreover, systematical analysis of explainability demonstrates that our framework can provide faithful explanations to facilitate the manual verification process.**

***Index Terms*—Table Interpretation, Explainability**

I. Introduction

In data management systems, table metadata, such as column types and relations between columns, have wide applications (e.g., Data Studio [1], Power BI [2], Tableau [3], and Trifacta [4]). Furthermore, it usually serves as the input of important downstream decision-making actions such as personally identifiable information (PII) detection in the digital marketing industry. Missing table metadata greatly might degenerate the performance of decision-making actions in data management systems, e.g., missing or false table metadata of PII may cause a severe privacy leakage [5]. However, in practice, it is common that tables miss table metadata [6]. Table interpretation aims to identify and uncover these metadata, including column type prediction and column relation prediction [6]–[8]. To ensure the quality of downstream decision-making actions, experts such as data stewards and marketers must manually verify the correctness of the predictions, which is very labor-intensive. To reduce the labour costs, providing explanations for TI's predictions is necessary as these explanations can help them do faster and more accurate verification. We refer to this as *explainable table interpretation*, which aims to provide faithful explanations and achieve high prediction performance for column type prediction and column relation prediction simultaneously.

Existing table interpretation (TI) approaches [6]–[10] have critical limitations in industry scenarios due to their lack of explainability and trustworthiness for manual verification. Lacking explainability incurs too much verification time for experts, which is expensive. Recently, EXACTA [5] has shown that producing explainability (e.g., evidence paths) for type prediction is of particular importance in industrial marketing scenarios. However, it asks crowd-sourced workers to manually assess whether those paths are suitable as explanations for the predictions. Crowds are challenging to scale [11], [12] and may yield relatively low-quality results if there is no proper quality control [13], [14]. To the best of our knowledge, existing work cannot provide any explanations for column relation prediction task. Motivated by these, we aim to explore the challenging explainable table interpretation problem along with developing an explainable TI method without additional manual intervention, which is a challenging attempt.

Challenge I: *How to produce faithful explanations without additional manual intervention?* In natural language applications, word attribution based on attention scores [15] is the predominant method for developing inherently explainable neural classifiers. Such methods explain the predictions locally using relevant words in input samples. However, such explanations were shown to be unreliable [16] and unfaithful [17], as to be demonstrated by Example I. Moreover, sentences and sentence pairs are compositional, explaining with higher-level compositional concepts (beyond individual word-level feature attributions) remains an open challenge [18]. Another known limitation is that the local explanations are limited to the input sample space, and often require additional methods [19] for providing global explanations, e.g., explaining the predictions via influential samples from the training set. SelfExplain [18] has incorporated both local and global explanations by constituent parsing. However, constituent parsing is unsuitable for table interpretation, because tables are more like sets (lack of syntactic structure) than normal sentences. This will be confirmed in the experiments to be presented in Section IV-C. Thus, producing faithful explanations without additional manual intervention is still challenging.

Example I: Figure 1(a) depicts a table from WikiTable, while Figure 1(b) gives an explainable table interpretation

[*]This work was partially done during the author's internship at Huawei.

[1]Explainability and interpretability are used interchangeably in some cases. We only use explainability in this paper to avoid confusion.

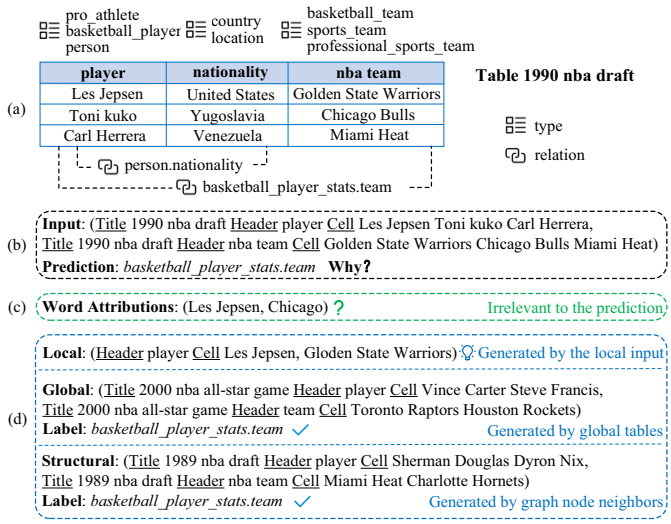


Fig. 1. (a) A table with the title "1990 nba draft" from WikiTable. (b) An example of column relation prediction. (c) Word-wise explanations from the word attribution based approach. (d) Multi-view explanations from ExplainTI.

example. Since the pre-trained transformer encoders take token sequences as an input, we transform the column pair into a sentence pair. Prediction denotes that there is a relation "basketball_player_stats.team" between the column "player" and column "nba team". Figure 1(c) shows that predominant word attribution methods explain the prediction locally using relevant words in the input sample. These explanations have two limitations: (i) word-wise explanations ("Les Jepsen" and "Chicago") are irrelevant to the prediction, in other words, they are useless for accelerating the manual verification; (ii) they are limited to the local view, considering from other views (e.g., global view) is necessary to produce faithful explanations. Moreover, we would like to emphasize that making predictions only using column values is doubtful. Take a column called "player" as an example. We can predict type "person", however, in order to predict "basketball_player' and "pro_athlete" must consider contextual information (e.g., column "nba team").

In addition to providing explanations, high prediction accuracy is also required. Considering tabular structure can be beneficial to the predictions. Nevertheless, recent BERT-based methods have the natural downside of implicit information loss in the tabular structure [8]. Motivated by this, our goal is to develop an explainable and effective TI framework based on pre-trained transformer encoders along with leveraging the contextual information provided by tabular structure, whose realization is non-trivial.

Challenge II: *How to aggregate contextual information effectively without losing explainability?* Since the columns in the table are semantically related, predicting based solely on the column values is an under-determined problem, as mentioned in Example I. It is widely acknowledged that contextual information can improve the TI performance [6], [8], [9], [20]. However, existing work based on pre-trained language models [6], [7], [9] fail to aggregate inter-table information because they only focus on aggregating components of a single table. Indeed, there are various implicit connections between columns of the same headers and titles across tables. Furthermore, aggregating intra-table information via serializing a single table is unexplainable. Therefore, aggregating both intra-table and inter-table contextual information effectively without losing explainability is still challenging.

To address above challenges, we propose ExplainTI, which explains table interpretation using multi-view explanations. Tables are firstly converted to sequences and column graphs, and then, ExplainTI fine-tunes a pre-trained transformer encoder to aggregate contextual information and provide multi-view explanations: (i) **Local Explanations** that aim to find the most relevant phrases (w.r.t. pairwise phrases) in columns (w.r.t. column pairs) for the predictions; (ii) **Global Explanations** that aim to find the most influential samples from the training data, as those samples tend to have similar labels to the input sample; and (iii) **Structural Explanations** that aim to provide explanations by considering the relations between columns. This module can not only aggregate both intra-table and inter-table contextual information but also provide explanations from the structural view via graph attention mechanism. Our contributions are summarized as follows:

- *Explainable TI framework.* We present a novel self-explaining table interpretation framework ExplainTI, which provides explanations for table interpretation predictions. To the best of our knowledge, ExplainTI is the first explainable framework in table interpretation.
- *Multi-view explanations.* Under the proposed framework, we develop multi-view explanations, including local explanations, global explanations, and structural explanations, in order to improve the explainable capability.
- *Effective TI performance.* We convert tables to sequences and column graphs, and then fine-tune a pre-trained transformer encoder to efficiently capture and integrate contextual information. Considerable experimental results demonstrate the superiority of ExplainTI on both Web tables and database tables.
- *Systematical explainability analysis.* We also systematically analyze the explainability of our framework in terms of sufficiency, plausibility, and trustability. The explanations from ExplainTI sufficiency reflect model predictions, and are more understandable for human annotators compared to strong explainable methods.

Outline. Section II presents the problem definition and overviews preliminaries. We introduce the proposed ExplainTI and its components in Section III. Section IV presents the experimental results. Finally, we discuss related work in Section V and conclude in Section V-B.

II. Preliminaries

In this section, we first formalize the problem of explainable table interpretation and then overview some background materials and techniques to be used in subsequent sections. Table I summarizes the symbols that are frequently used throughout this paper.

TABLE I
SYMBOLS AND DESCRIPTION.

Symbol	Description
$\mathbf{T}$	A set of tables
T	A relational table $T = (c_1, c_2, \ldots, c_n)$
c_i	A column $c_i = (v_1^i, v_2^i, \ldots, v_m^i)$
h_i	The header of a column c_i
p	The title of a table
$\mathbf{X}$	The input sequences
$\mathbf{E}$	The embeddings
$\mathcal{M}$	The transformer Encoder
$\mathcal{Z}$	Explainable concepts
Q	The embedding store
$\mathcal{N}(c)$	Neighbors of column c
RS	The relevance score for local explanations
IS	The influence score for global explanations
AS	The attention score for structural explanations
$\mathcal{C}_{\text{type}}$, $\mathcal{C}_{\text{rel}}$	Column type labels and column relation labels
D_{type}, D_{rel}	The training sets of type prediction and relation prediction
$\mathcal{G}_t$, $\mathcal{G}_r$	Column graph and column pair graph

A. Problem Formulation

We consider the explainable table interpretation problem in the framework of neural text classification. There are two explainable table interpretation problems, including explainable column type and column relation predictions. The goal of the explainable column type prediction is to predict the type of a column from among a fixed set of predefined types along with explaining the prediction. For the explainable column relation prediction, our goal is to predict the relation of each pair of columns and explain the prediction. These two problems are formulated as follows.

Definition 1: (Explainable Column Type Prediction). Given a table T and a set of column type labels $\mathcal{C}_{\text{type}}$, *explainable column type prediction* is, for every column $c_i \in T$, to predict a type $y \in \mathcal{C}_{\text{type}}$ along with explanations $\mathcal{Z}$ for the prediction.

Definition 2: (Explainable Column Relation Prediction). Given a table T and a set of column relation labels $\mathcal{C}_{\text{rel}}$, *explainable column relation prediction* is to predict a relation $y \in \mathcal{C}_{\text{rel}}$ for every column pair along with explanations $\mathcal{Z}$ for the prediction.

B. Pre-trained Language Models

Pre-trained language models (LMs) (e.g., BERT [21] and RoBERTa [22]) have demonstrated powerful semantic expression abilities, which can support lots of downstream tasks (e.g., classification and question answering). Concretely, we can plug appropriate inputs and outputs into a pre-trained LM based on the specific task and then fine-tune all the model's parameters end-to-end.

Intuitively, the column type prediction problem can be treated as a sentence classification task [6]. Given a column $c_i \in T$, we can transform it into a sentence $\mathcal{S}(c_i)$. A sentence $\mathcal{S}(c_i)$ is denoted by $\mathcal{S}(c_i) ::= \langle$[CLS] Title p Header h_i Cell v_1^i... v_m^i [SEP]$\rangle$, where [CLS] and [SEP] are special tokens used to mark the beginning and end of a sequence. Similarly, we can treat column relation prediction problem as a sentence pair classification task. A column pair (c_i, c_j) can be serialized as a pairwise sentence $\mathcal{S}(c_i, c_j) ::= \langle$[CLS] Title p Header h_i Cell v_1^i ... v_m^i [SEP] Title p Header h_j Cell v_1^j ... v_m^j [SEP]$\rangle$. [CLS] is utlized to store the classification output signals during the fine-tuning of LMs.

III. THE PROPOSED APPROACH

In this section, we first introduce the ExplainTI framework, which explains table interpretation using multi-view explanations. It includes three modules: Local Explanations (LE), Global Explanations (GE), and Structural Explanations (SE). Then, we describe the end-to-end training procedure and optimization objectives. Finally, we discuss the technical novelty of our approach.

A. Framework Overview

Let $\mathcal{M}$ be a pre-trained transformer model (e.g., BERT) that maps $\mathbf{X} \rightarrow \mathbf{E}$, where $\mathbf{X}$ are the input sequences and $\mathbf{E}$ are the output embeddings. ExplainTI builds into $\mathcal{M}$, and provides a set of explanations $\mathcal{Z}$ for the predictions.

As illustrated in Figure 2, in the first step, given a set of tables $\mathbf{T}$, tables are converted to sequences and a column graph for explainable column type prediction, while tables are converted to sentence pairs and a column pair graph for explainable column relation prediction. In the second step, $\mathcal{M}$ is trained end-to-end to produce explanations $\mathcal{Z}$ from multi-view using the above sequences and graphs. From the local view, we compute the most relevant phrases (w.r.t. pairwise phrases) in columns (w.r.t. column pairs) for the predictions to produce local explanations (LE). From the global view, global explanations (GE) are the most influential samples from training data, which tend to have similar labels with the input sample. From the structural view, we aggregate both intra-table and inter-table contextual information using neighbors sampled from graphs, and then provide the most influential neighbors, namely, structural explanations (SE).

We treat table interpretation as the downstream objective to fine-tune $\mathcal{M}$. Specifically, for each sentence or sentence pair in $\mathbf{X}$, we filter the CLS embedding of $\mathbf{E}$ by an output layer to obtain

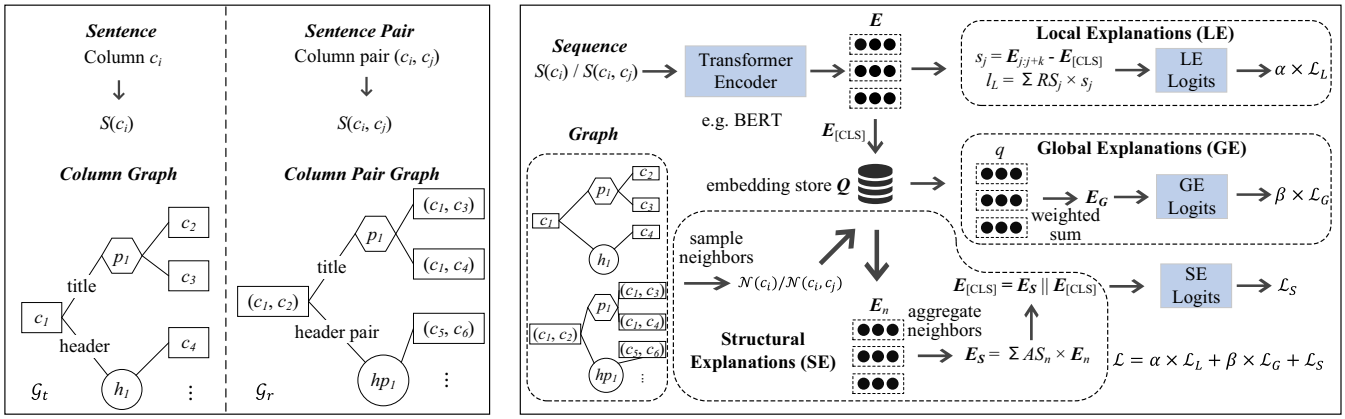
$$logits = \sigma(\mathbf{W} \times \mathbf{E}_{[\mathbf{CLS}]} + \mathbf{b}) \tag{1}$$

Here, $\mathbf{W} \in \mathbb{R}^{d \times c}$, $\mathbf{b}$ is the bias for the linear layer, d is the dimension of $\mathbf{E}_{[\mathbf{CLS}]}$, and c is the number of labels. $\sigma(\cdot)$ denotes sigmoid function in multi-label prediction task or softmax function in multi-class prediction task.

B. Local Explanations (LE)

We compute relevance scores for phrases (pairwise phrases) in columns (column pairs) for local explanations. Recent approaches [18], [23] assign relative importance scores to input features through activation differences. Motivated by this, we adopt phrases (pairwise phrases) with fixed window size continuous tokens as explainable concepts. Thereafter, we learn the relevance score of each concept to the final label distribution via their activation differences, which is a variant of the attention mechanism. Based on relevance scores, we can focus on the most relevant concept to the prediction.

Algorithm 1 shows how we learn the attribution of each concept to the final label distribution. Given a sample x in $\mathbf{X}$,



(a) Convert tables to sequences and graphs

(b) Multi-view training

Fig. 2. The proposed ExplainTI framework.

we first build a representation of the input without contribution of each concept. The concepts are phrases for column type prediction task, $t_j = \text{mean}(\mathbf{E}_{j:j+k}) - \mathbf{E}_{[\text{CLS}]}$. The concepts are pairwise phrases for column relation prediction task, $t_j = (\text{mean}(\mathbf{E}_{j:j+k}) + \text{mean}(\mathbf{E}_{j^*:j^*+k}))/2 - \mathbf{E}_{[\text{CLS}]}$, where j and j^* are the starting position of the windows, the embedding of each concept is obtained via a token-wise mean pooling function, k is the windows size, and t_j is the representation of the input without contribution of each concept. Then we use t_j to score the labels:

$$s_j = \sigma(\mathbf{W}_l \times t_j + \mathbf{b}_l) \tag{2}$$

where $t_j \in \mathbb{R}^d$, $s_j \in \mathbb{R}^c$, $\mathbf{W}_l \in \mathbb{R}^{d \times c}$. Here, s_j signifies a label distribution without the contribution of each concept. Using this, the relevance score of each concept for the final prediction can be computed:

$$RS_j = \frac{\text{KL}(s_j, logits)}{\sum_{j'} \text{KL}(s_{j'}, logits)}, \tag{3}$$

where RS_j is the relevance score, KL represents Kullback-Leibler Divergence that can measure the closeness of two distributions. Some recent studies [24], [25] use functions over distributions (e.g., KL) for various applications. Here, we employ KL in a different way (i.e., computing relevance scores for local explanations). The per-batch time complexity of the local explanations is $O(|B| \times n)$ and $O(|B| \times n^2)$ for type prediction and relation prediction, respectively, where $|B|$ is batch size, and n is the sequence length. This process can be easily parallelized on the GPU. We give an illustrative example of local explanations in Figure 1(d), where ExplainTI-Local gives the top relevant pairwise phrases (i.e., the top local explanation) for the column relation prediction.

Algorithm 1: Local Explanations

Input: a mini-batch of samples B_{task}, a window size k
Output: local logits l_L, relevance scores RS

```
for i ← 0 to len(B_task) do
    tot ← 0
    x ← B_task,i
    E ← M(x)
    logits ← Linear(E)
    if task = type then
        for j ← 0 to len(x) − k do
            // Representation of the sample
               without each window
            window ← x_[j:j+k]
            t_j ← E_window − E_[CLS]
            s_j ← Linear(t_j)
            tot ← tot + KL(s_j, logits)
        l_L,i ← zero logits
        for j ← 0 to len(x) − k do
            // Calculate the relevance score
            RS_i,j ← KL(s_j, logits) / tot
            // Calculate the local logits
            l_L,i ← l_L,i + RS_i,j × s_j
    else
        for j ← 0 to len(x)/2 − k do
            for j* ← len(x)/2 to len(x) − k do
                window ← x_[j:j+k,j*:j*+k]
                t_jj* ← E_window − E_[CLS]
                s_jj* ← Linear(t_j)
                tot ← tot + KL(s_jj*, logits)
        l_L,i ← zero logits
        for j ← 0 to len(x)/2 do
            for j* ← len(x)/2 + 1 to len(x) do
                RS_i,jj* ← KL(s_jj*, logits) / tot
                l_L,i ← l_L,i + RS^tot_i,jj* × s_jj*
return l_L, RS
```

C. Global Explanations (GE)

Global explanations aim to explain each data sample by providing a set of K influential samples from the training data. Influential samples tend to have similar labels to the input sample. Experts can understand the correctness of predictions via these influential samples. An illustrative example of global explanations is shown in Figure 1. Intuitively, the most influential sample from the global view has the same label as the prediction.

Algorithm 2 shows the process of global explanations. We first build an *embedding store* Q which holds all the embeddings of the training data. Given a model $\mathcal{M}$, we

Algorithm 2: Global Explanations

Input: a mini-batch of samples B_{task}, an embedding store Q, a hyperparameter K
Output: global logits l_G, influence scores IS

for $i \leftarrow 0$ **to** $\text{len}(B_{task})$ **do**
  $x \leftarrow B_{task,i}$
  $\mathbf{E} \leftarrow \mathcal{M}(x)$
  // Retrieve the top-K influential samples
  $q \leftarrow \text{IndexHNSW}(Q, \mathbf{E}_{[\mathbf{CLS}]}, K)$
  $tot \leftarrow 0$
  for $j \leftarrow 0$ **to** K **do**
    $tot \leftarrow tot + \exp\left(\cos\left(\mathbf{E}_{[\mathbf{CLS}]}, q_j\right)\right)$
  $\mathbf{E}_{i,G} \leftarrow zero\ embeeding$
  for $j \leftarrow 0$ **to** K **do**
    // Calculate the influential score
    $IS_{i,j} \leftarrow \frac{\exp\left(\cos\left(\mathbf{E}_{[\mathbf{CLS}]}, q_j\right)\right)}{tot}$
    // Calculate the global embedding
    $\mathbf{E}_{i,G} \leftarrow \mathbf{E}_{i,G} + IS_{i,j} \times q_j$
$l_G \leftarrow \text{Linear}(\mathbf{E}_G)$
return l_G, IS

Algorithm 3: Column Graph Construction

Input: a set of tables $\mathbf{T}$
Output: a column graph $\mathcal{G}_t$, a column pair graph $\mathcal{G}_r$

// Initialize the graph $\mathcal{G}_t$ and $\mathcal{G}_r$
$\mathcal{G}_t, \mathcal{G}_r \leftarrow \varnothing$
for $T \in \mathbf{T}$ **do**
  $p \leftarrow$ find the title of T
  $\mathcal{G}_t$.addNode(p), $\mathcal{G}_r$.addNode(p)
  // Add the nodes and edges to $\mathcal{G}_t$
  for $c_i \in T$ **do**
    $h_i \leftarrow$ find the header of c_i
    $\mathcal{G}_t$.addNode(c_i)
    $\mathcal{G}_t$.addEdge(c_i, title, p)
    $\mathcal{G}_t$.addEdge(c_i, header, h_i)
  // Add the nodes and edges to $\mathcal{G}_r$
  for $(c_i, c_j) \in T$ **do**
    $(h_i, h_j) \leftarrow$ the header pair of (c_i, c_j)
    $\mathcal{G}_r$.addNode((c_i, c_j))
    $\mathcal{G}_r$.addEdge((c_i, c_j), title, p))
    $\mathcal{G}_r$.addEdge((c_i, c_j), header pair, (h_i, h_j))
return $\mathcal{G}_t$, $\mathcal{G}_r$

represent each sample from the training data using the CLS embedding $\mathbf{E}_{[\mathbf{CLS}]}$. Q is denoted by a set of $\{q\}_{1:N_Q}$, where N_Q is the number of training samples. Q is initialized at the beginning of fine-tuning $\mathcal{M}$, and then will be updated after every fixed number of training steps.

For any input x with embedding $\mathbf{E}_{[\mathbf{CLS}]}$, we retrieve the top-$K$ influential samples $\{q\}_{1:K}$ by influence scores:

$$IS\left(q \mid \mathrm{x}\right) = \frac{\exp\cos\left(\mathbf{E}_{[\mathbf{CLS}]}, q\right)}{\sum_{q'} \exp\cos\left(\mathbf{E}_{[\mathbf{CLS}]}, q'\right)} \tag{4}$$

where cos denotes cosine similarity function. The efficiency of global explanations depends on the retrieve process. Hence, we accelerate the retrieve process effectively via employing IndexHNSW [26] implemented by faiss [27]. The per-batch search time complexity of global explanations is $O(|B| \times log(N_Q))$ in principle, where $|B|$ is batch size.

D. Structural Explanations (SE)

Inspired by the graph structure's powerful capturing ability of semantics, we propose a structural explanations module to aggregate contextual information. It consists of a column graph construction approach for transforming tables from the relational format to the graph structure, and then, it learns the structural features and produces explanations from structural view via graph attention mechanism. Especially, the search space of the structural explanations is a set of neighbor samples, which is more fine-grained than the search space of the global explanations (i.e., all training samples).

*1) **Column graph construction**:* Graph construction techniques have been presented in the existing text classification work, such as TextGCN [28] and BertGCN [29]. Nonetheless, several drawbacks (i.e., large-scale graph embedding and lack of explainability) restrict the scope of using these graph construction methods to perform TI in real-world scenarios.

The above limitations motivate us to design a relatively lightweight but highly effective graph construction method. The table titles and column headers (header pairs) can greatly affect the TI predictions, and can be used to correlate contextual information. For example, the fact that the columns (column pairs) in a table having the same table title and columns (column pairs) with the same headers (header pairs) among multiple tables are related. These implicit connections between columns (column pairs) can be important for learning a prediction model. Therefore, we extract the titles, headers (header pairs), and corresponding columns (column pairs) for column graph construction. We treat columns (column pairs) as a whole, and thus, the constructed graphs are relatively lightweight. We describe the task-specific column graph construction procedure, with its pseudo-code presented in Algorithm 3. Given a set T of tables, it initializes an empty column graph $\mathcal{G}_t$ and an empty column pair graph $\mathcal{G}_r$ (Line 1). Then, it iteratively adds nodes and edges to $\mathcal{G}_t$ and $\mathcal{G}_r$ via implicit connections relying on titles and headers (header pairs) (Lines 2-14). Finally, the column graph $\mathcal{G}_t$ and the column pair graph $\mathcal{G}_r$ are returned to be utilized in column type prediction and column relation prediction, respectively. The time complexity of column graph construction is $O(|\mathbf{T}| \times |T|)$, where $|\mathbf{T}|$ is the number of tables, $|T|$ is the number of columns in a table, and $|T| \ll |\mathbf{T}|$.

*2) **Aggregating contextual information**:* Inspired by the previous work [30], we generate embeddings by sampling and aggregating features from nodes' local neighborhoods, which is inductive and efficient.

Neighbors sampling. Titles and headers (w.r.t. header pairs) can serve as the bridge connecting two columns (w.r.t. column pairs). Taking a column graph $\mathcal{G}_t$ in Figure 2 as an example, column c_1 has title p_1 and header h_1, c_2 (w.r.t. c_4) connects c_1 via its title p_1 (w.r.t. header h_1), which can enrich column c_1's features. We build upon this intuition to perform neighbors sampling between a column (column pair) and its neighbors. Formally, given a column c_i or a column pair (c_i, c_j), we uniformly sample a set of 2-hop neighbors denoted by $\mathcal{N}_i$ or

$\mathcal{N}_{ij}$. Specifically, considering the computational efficiency, a certain number of neighbors are sampled for each node in graphs. If the number of neighbors is less than the sampling size r, the sampling method with replacement is used until r neighbors are sampled.

Information aggregation. A straightforward way to consider contextual information is by applying a pooling operator (e.g., mean pooling) on their embeddings. However, different neighbors have various contributions to the predictions, and they should be considered differently. Graph attention network [31] can capture the importance of neighbors. Moreover, attention is a valuable tool to explain model decisions in recent work [32], [33]. Therefore, to aggregate neighbors without losing explainability, we exploit the idea of the graph attention network. Algorithm 4 shows our structural explanations based on the idea of using graph attention mechanism.

Considering a column c_i, we use $\mathcal{N}_i = \{c_n | c_n \in \mathcal{G}_t\}$ to denote its neighbors. We implement the attention score AS via dot-product attention mechanism, which is formulated as follows:

$$AS_n = \frac{\exp\left(\mathbf{E}_i \cdot \mathbf{E}_n\right)}{\sum_{c_{n'} \in \mathcal{N}_i} \exp\left(\mathbf{E}_i \cdot \mathbf{E}_{n'}\right)} \tag{5}$$

where $\mathbf{E}_i$ is the CLS embedding of c_i, and $\mathbf{E}_n$ are embeddings of c_n retrieved from the embedding store Q. Note that, columns and column pairs have the same process for aggregating information. Hereafter, we aggregate information:

$$\mathbf{E}_s = \sum_{c_n \in \mathcal{N}_i} AS_n \mathbf{E}_n \tag{6}$$

where $\mathbf{E}_s$ is the contextual embedding of c_i. The per-batch time complexity of information aggregation is $O(|B| \times r)$, where $|B|$ is batch size, and r is the sample size.

Note that, the structural explanations module plays two roles in our framework: It directly contributes to the predicted results and explains the predictions from the structural view.

E. Training

1) Multi-task training: In the training phase, following the previous study [6], ExplainTI fine-tunes a pre-trained LM using two different training data and two different objectives. The training process of ExplainTI is presented in Algorithm 5. Concretely, we switch every epoch, and update the parameter for different objectives using different optimization schedulers (Lines 5-6). This enables ExplainTI to handle imbalanced training data for different tasks naturally.

2) Multi-view training: During our training, we regularize the loss via three views by optimizing their output for the end-task as well. The details are provided as below.

For local view, we compute a weighted aggregated representation over s_j: $l_L = \sum_j RS_j \times s_j$ (Line 8), and then compute the log-likelihood loss (Line 9):

$$\mathcal{L}_L = -\sum_{c=1}^{C} y_c \log\left(l_L\right) \tag{7}$$

For global view, we get the global representation by aggregating all the retrieved $\{q\}_{1:K}$ as a weighted sum: $\mathbf{E}_G = \sum_{k=1}^{K} IS\left(q_k \mid \mathrm{x}\right) q_k$, and compute $l_G = \sigma(\mathbf{W}_g \times \mathbf{E}_G + \mathbf{b}_g)$ (Line 8), where $\mathbf{W}_g \in \mathbb{R}^{d \times c}$, $\mathbf{b}_g$ is the bias, $IS\left(q_k \mid \mathrm{x}\right)$ is the influence score of q_k. Next, the loss is computed as (Line 10):

$$\mathcal{L}_G = -\sum_{c=1}^{C} y_c \log\left(l_G\right) \tag{8}$$

For structural view, we fuse sentence features and structural features to obtain $\mathbf{E}^*_{[\mathrm{CLS}]} = \mathbf{E}_s || \mathbf{E}_{[\mathrm{CLS}]}$, where $||$ is the concatenation operation. Then $\mathbf{E}^*_{[\mathrm{CLS}]}$ is used for the final classification (Line 8):

$$logits = \sigma(\mathbf{W}_s \times \mathbf{E}^*_{[\mathbf{CLS}]} + \mathbf{b}_s) \tag{9}$$

Algorithm 4: Structural Explanations

Input: a mini-batch of samples B_{task}, an embedding store Q, sample size r, a column graph $\mathcal{G}_{task}$
Output: final logits $logits$, attention scores AS

for $i \leftarrow 0$ **to** $\mathrm{len}(B_{task})$ **do**
  $x \leftarrow B_{task,i}$
  $\mathbf{E} \leftarrow \mathcal{M}(x)$
  `// Sample the neighbors of x`
  $\mathcal{N} = \mathrm{Sample}(\mathcal{G}_{task}, x, r)$
  $E_{\mathcal{N}} \leftarrow$ from Q
  `// Aggregate the information of neighbors`
  $tot \leftarrow 0$
  for $j \leftarrow 0$ **to** r **do**
    $tot \leftarrow tot + \exp\left((\mathbf{E}_{[\mathbf{CLS}]} \cdot \mathbf{E}_j)\right)$
  $\mathbf{E}_{i,s} \leftarrow zero\ embedding$
  for $j \leftarrow 0$ **to** r **do**
    `// Calculate the attention score`
    $AS_{i,j} \leftarrow \frac{\exp\left((\mathbf{E}_{[\mathbf{CLS}]}, \mathbf{E}_j)\right)}{tot}$
    `// Calculate the contextual embedding`
    $\mathbf{E}_{i,s} \leftarrow \mathbf{E}_{i,s} + AS_{i,j} \times \mathbf{E}_j$
    $\mathbf{E}^*_{i,[\mathrm{CLS}]} = \mathbf{E}_{i,s} || \mathbf{E}_{[\mathrm{CLS}]}$
$logits \leftarrow \mathrm{Linear}(\mathbf{E}^*_{[\mathrm{CLS}]})$
return $logits, AS$

Algorithm 5: Training Process of ExplainTI

Input: number of training epochs N_{epoch}; training sets $\{D_{type}, D_{rel}\}$
Output: a model $\mathcal{M}_\theta$

`// Initialize model weights`
Initialize $\mathcal{M}_\theta$ with a pre-trained language model;
`// Serialize tables in training sets`
for $task \in \{type,\ rel\}$ **do**
  $D_{task} \leftarrow \{\mathrm{serialize}(T)\ for\ T \in D_{task}\}$
for $ep \leftarrow 1$ **to** N_{epoch} **do**
  `// Switch task for each epoch`
  for $task \in \{type,\ rel\}$ **do**
    Split D_{task} into mini-batches $\{B_1, ...B_k\}$;
    for $B \in \{B_1, ...B_k\}$ **do**
      `// Forward`
      $l_L, l_G, logits \leftarrow \mathcal{M}(B)$
      `// Calculate the loss`
      $\mathcal{L}_L \leftarrow Loss(l_L, B)$
      $\mathcal{L}_G \leftarrow Loss(l_G, B)$
      $\mathcal{L}_S \leftarrow Loss(logits, B)$
      $\mathcal{L} \leftarrow \mathcal{L}_S + \alpha \times \mathcal{L}_L + \beta \times \mathcal{L}_G$
      `// Optimize the model`
      $\mathcal{M}_\theta \leftarrow \mathrm{back\text{-}propagate}(\mathcal{M}_\theta, \mathcal{L})$
return *the best model* $\mathcal{M}_\theta$

where $\mathbf{W}_s \in \mathbb{R}^{2d \times c}$, $\mathbf{b}_s$ is the bias of the classification layer. Then the loss of structural view is computed (Line 11):

$$\mathcal{L}_S = -\sum_{c=1}^{C} y_c \log(logits) \tag{10}$$

To train the model, we optimize for the following joint loss (Line 12):

$$\mathcal{L} = \alpha \times \mathcal{L}_L + \beta \times \mathcal{L}_G + \mathcal{L}_S \tag{11}$$

Here, α and β are regularization hyperparameters. We use Binary Cross-Entropy loss for multi-label prediction and Cross-Entropy loss for multi-class prediction.

F. Discussion

Our work extends SelfExplain into tabular data; however, the extension is non-trivial. Our work has significantly improved the methodology of SelfExplain and brought the following technical novelty.

- **Normal sentences vs. Tabular data**. SelfExplain is designed for normal sentences, which provides explanations by the means of constituent parsing. However, constituent parsing cannot be used in table interpretation, because tables are more like sets (lack of syntactic structure) than normal sentences (Challenge I, Section I, Page 1). By contrast, our method is designed for tabular data, which can produce explanations in a novel way: tables are firstly converted to sequences and column graphs, and then, a pre-trained transformer encoder is fine-tuned to aggregate contextual information and provide multi-view explanations without additional manual intervention.
- **Structural explanations.** SelfExplain ignores the important structural view, which notably impacts its effectiveness and explainability. To fill this gap, we propose to adopt the structural view that provides structural explanations by considering the relations between columns. This module contributes to both effectiveness and explainability, as evaluated in Section IV-C and Section IV-D.

IV. Experiments

In this section, we experimentally evaluate our proposed ExplainTI in terms of both table interpretation performance and the explainability on two publicly available datasets. In our experiments, we consider two transformer encoder configurations as our base models: BERT [21] and RoBERTa [22]. We aim at answering the following research questions:

- **RQ1**: How does the proposed method perform compared with the state-of-the-art methods for predicting column type and pairwise relation between columns?
- **RQ2**: Does our method have the ability to provide good explainability for table interpretation?
- **RQ3**: How does each module affect the overall performance of the model?
- **RQ4**: How is the efficiency of ExplainTI?
- **RQ5**: What is the impact of different parameters on the sensitivity of ExplainTI?

A. Experimental Setup

Datasets. We use two real-world and large-scale benchmark datasets from different types for evaluation. The statistics of datasets are summarized in Table II. The WikiTable dataset [7] contains 462,676 Web tables collected from Wikipedia, which are annotated with column types and pairwise relations. The dataset defines 255 column types and 121 pairwise relation types. We use the same train/valid/test splits as TURL [7].

For the database table, the GitTable [34] dataset is the first large-scale relational table corpus with topical coverage and content structurally different than tables extracted from HTML pages. The dataset is used for the column type prediction task. Columns in GitTable are annotated semi-automatically with more than 2K different semantic types from Schema.org [35] and DBpedia [36]. Note that, how to construct datasets without manual intervention is outside the scope of our work. We use its subset organism, and split it into train, valid, and test using the radio of 8:1:1.

Baselines. We mainly use the following three categories of baselines, including specially designed methods, tabular data representation learning methods, and explainable text classification approaches. Specially designed methods are designed for column type prediction or pairwise column relation prediction. Note that, EXACTA [5] is an explainable column type prediction method. However, it requires additional manual intervention (i.e., crowdsourced-based large-scale path annotations), which is costly. Thus, we omit it in the following discussions, as we focus on producing explanations without any manual intervention. Besides, we compare the proposed ExplainTI against the state-of-the-art tabular data representation learning methods. To verify explainability, we compare ExplainTI with several widely used explainable text classification approaches, which have presented promising performance.

- *Specially designed methods*: Sherlock [37] is a column type prediction deep model that utilizes multiple feature sets extracted from each column. Sato [10] is a hybrid machine learning model based on Sherlock with additional topic modeling components.
- *Tabular data representation learning methods*: TaBERT [9] is a pre-trained encoder for a joint understanding of textual and tabular data. TURL [7] uses a structure-aware Transformer encoder to model the row-column structure. Doduo [6] is a unified table interpretation method based on pre-trained language models to support multi-task learning. TCN [8] makes use of both information within the table and across multiple tables from similar domains to predict column types and pairwise column relations.
- *Explainable text classification approaches*: We compare our framework against three explainable methods: (i) Saliency map [38] for local word attributions, (ii) Influence functions [19] that provides explanations from global view and (iii) SelfExplain [18] that is the state-of-the-art method for both local explanations and global explanations.

Implementation details. We implement ExplainTI in PyTorch

TABLE II
STATISTICS OF THE DATASETS.

Name	type	# tables	Avg. # rows	Avg. # cols	# labels
WikiTable	Web tables	462,676	12.4	1.7	255/121
GitTable	database tables	12,200	152.9	4.0	1,141

and Transformers library. We currently support 2 pre-trained language models: BERT [21] and RoBERTa [22]. We use the base variant of each model in all our experiments. We further apply the half-precision floating-point (fp16) optimization to save the GPU memory usage and running time. In all the experiments, the max sequence length is set to 64 in the token level; the learning rate is set to 5e-5 with a linearly decreasing learning rate schedule; the batch size is set to 160; the K is set to 10; the size of the windows in local explanations (LE) is set to 8; and the sampling size in SE is set to 16. We use AdamW as the optimizer for training, and fix the epoch to be 40. We update the embedding store Q every 5 epochs. We tune the hyperparameters by doing a search α and β in $\{0.05, 0.10, 0.15, 0.20, 0.25, 0.50\}$, and select the one with the best performance. We select the epoch with the highest F1-weighted on the validation set, and report results on the test set. All the experiments were conducted on a machine with an NVIDIA A100 GPU with 40GB of memory. The code and all datasets are available at https://github.com/ZJU-DAILY/ExplainTI. We implement baselines following the original papers and corresponding public codes.

- Sherlock [37], Sato [10]: We implement Sherlock and Sato following the original papers and public code_1 and code_2. As they are originally designed for predicting semantic type of columns, we concatenate the embeddings of subject and object pair of columns to predict the relation.
- TaBERT [9]: We implement TaBERT following the origin paper and public code.
- TURL [7]: We implement this method according to the original paper and public code. Following TCN [8], we set the embedding of linked entity in TURL the same as its cell since they are not provided as inputs in our case.
- Doduo [6]: We implement this method according to the origin paper and public code.
- TCN [8]: We implement TCN according to the origin paper.
- Saliency Map [38], Influence Functions [19]: We implement these methods following the origin papers and public code.
- SelfExplain [18]: We implement SelfExplain according to the origin paper and public code.

Evaluation metrics. Following related studies [7], [8], [10], [37], we treat predicting column type and pairwise relation between columns as a classification task. We leverage three widely-used classification metrics, namely, F1-micro, F1-macro, and F1-weighted.

B. Experiment on Table Interpretation (RQ1)

We first evaluate the table interpretation performance of our framework compared to the above baselines. The benchmark results of all methods across the datasets are reported in Table III.

ExplainTI vs. specially designed methods. As observed, ExplainTI significantly outperforms all the specially designed methods. Concretely, the performance of Sherlock and Sato are poor, since they do not utilize the recent advances in deep language models.

ExplainTI vs. tabular data representation learning methods. Tabular data representation learning methods outperform all the specially designed methods. The potential reason is that the transformer architecture effectively captures sequence patterns. As we can see, the performance of ExplainTI is superior to the SOTA methods. Particularly, TCN achieves poor performance on GitTable. The reason is that GitTable is database tables that are more heterogeneous than the text-heavy Web tables [34].

ExplainTI vs. explainable text classification approach. We compare our framework against the state-of-the-art inherently explainable text classification approach SelfExplain. It was originally designed for a single input sentence, and we extend it to support TI tasks. Here, Saliency map and Influence functions are omitted because they are post-hoc explainable methods that do not affect model performance. We can observe that ExplainTI outperforms SelfExplain in all cases. Moreover, SelfExplain has considerable performance in most cases, meaning that treating TI tasks as text classification tasks is suitable.

Overall, the proposed ExplainTI can consistently outperform baselines on all metrics across two datasets. For column type prediction task, ExplainTI scores an average F1 of 0.901 on WikiTable which is +0.164 relatively over competitive baselines, and scores an average F1 of 0.944 on GitTable which is +0.200 over baselines. For column relation prediction task, ExplainTI scores an average F1 of 0.924 on WikiTable which is +0.108 over baselines. This justifies the effectiveness of ExplainTI against baselines.

C. Experiment on Explainability (RQ2)

A *good* model explanation should be (i) relevant to the current input and predictions, and (ii) understandable to humans [39]–[41]. Following the previous study [18], we evaluate the explanations from three diverse criteria: sufficiency, plausibility, and trustability.

Do ExplainTI explanations reflect predicted labels? *Sufficiency* aims to evaluate whether model explanations alone are highly indicative of the predictions [18]. In other words, the sole explanations, without the input, can be sufficient for predicting labels. FRESH [41] is a framework to evaluate sufficiency of explanations. Following FRESH, we train a pre-trained language model (e.g., RoBERTa [22]) to perform a task using only extracted explanations without the input samples. An explanation that achieves high performance using this classifier is sufficient to reflect the origin model predictions. Table IV reports the corresponding results.

ExplainTI-LE denotes the local explanations of ExplainTI, ExplainTI-GE represents the global explanations of Ex-

TABLE III
TABLE INTERPRETATION PERFORMANCE ON WIKITABLE AND GITTABLE.

Method	Column type (WikiTable)			Column relation (WikiTable)			Column type (GitTable)		
	F1-micro	F1-macro	F1-weighted	F1-micro	F1-macro	F1-weighted	F1-micro	F1-macro	F1-weighted
Sherlock	0.662	0.326	0.640	0.757	0.520	0.722	0.740	0.413	0.723
Sato	0.673	0.335	0.659	0.710	0.346	0.696	0.857	0.475	0.854
TaBERT	0.841	0.580	0.830	0.920	0.842	0.915	0.903	0.453	0.875
TURL	0.920	0.715	0.915	0.920	0.847	0.918	0.919	0.728	0.915
Doduo	0.922	0.750	0.921	0.915	0.839	0.912	0.903	0.638	0.898
TCN	0.928	0.671	0.922	0.933	0.849	0.929	0.723	0.337	0.680
SelfExplain	0.875	0.513	0.887	0.910	0.819	0.914	0.962	0.680	0.953
ExplainTI-BERT	**0.944**	**0.815**	**0.944**	**0.941**	**0.891**	**0.941**	0.982	0.863	0.980
w/o LE	0.942	0.811	0.942	0.938	0.878	0.938	0.982	0.858	0.980
w/o GE	**0.944**	**0.815**	**0.944**	0.939	0.890	0.939	0.982	0.863	0.980
w/o SE	0.932	0.797	0.930	0.935	0.875	0.933	0.982	0.863	0.980
w PP	0.942	0.811	0.942	0.941	0.895	0.941	0.982	0.856	0.980
ExplainTI-RoBERTa	0.941	0.811	0.940	0.939	0.890	0.940	**0.983**	**0.873**	**0.982**
w/o LE	0.941	0.809	0.940	0.934	0.874	0.933	**0.983**	0.871	0.981
w/o GE	0.940	0.808	0.940	0.939	0.886	0.938	**0.983**	**0.873**	**0.982**
w/o SE	0.929	0.781	0.928	0.937	0.878	0.936	0.982	0.857	0.980
w PP	**0.944**	**0.815**	0.943	0.939	0.888	0.939	**0.983**	0.864	**0.982**

TABLE IV
MODEL PREDICTIVE PERFORMANCES. WE USE EXPLANATIONS PROVIDED BY EXPLAINTI-ROBERTA FOR THIS ANALYSIS. ALL THE METHODS USE AT MOST TOP-K EXPLAINABLE CONCEPTS. K=10 FOR SALIENCY MAP DUE TO ITS EXPLANATIONS ARE SHORT, K=1 FOR EXPLAINTI-GE AND EXPLAINTI-SE, AND K=3 FOR OTHER METHODS.

Method	Column type (WikiTable)			Column relation (WikiTable)			Column type (GitTable)		
	F1-micro	F1-macro	F1-weighted	F1-micro	F1-macro	F1-weighted	F1-micro	F1-macro	F1-weighted
Saliency Map	0.092	0.005	0.084	0.019	0.005	0.019	0.254	0.110	0.320
Influence Functions	0.200	0.009	0.135	0.105	0.006	0.052	0.067	0.025	0.033
SelfExplain-Local	0.488	0.220	0.472	0.261	0.089	0.226	0.571	0.154	0.578
SelfExplain-Global	0.169	0.005	0.139	0.092	0.002	0.019	0.009	0.001	0.009
ExplainTI-LE	0.725	0.345	0.722	0.576	0.296	0.555	0.622	0.262	0.590
ExplainTI-GE	**0.934**	**0.760**	**0.934**	0.910	0.823	0.909	**0.959**	**0.836**	**0.953**
ExplainTI-SE	0.906	0.661	0.901	**0.932**	**0.887**	**0.933**	0.758	0.421	0.739

plainTI, and ExplainTI-SE denotes the structural explanations of ExplainTI. From Table IV, we observe that all explanations from ExplainTI achieve high predictive performance compared to all the baselines across both Web tables and database tables. TI tasks are more intractable (too many labels and long text) than normal text classification so that all baselines achieve poor performance. As mentioned in Section I, the SOTA explainable approach SelfExplain cannot deal with our task well. Additionally, ExplainTI-GE and ExplainTI-SE are comparable to the full-text (an explanation that uses all of the input samples) using only top-1 explanation in most cases. The results indicate that GE from the global view and SE from the structural view are very relevant to reflect predicted labels. To further verify the sufficiency of ExplainTI-LE, we compare it with a random window selection strategy, which selects local explanations randomly instead of using relevance scores. As shown in Figure 3, our ExplainTI-LE outperforms the random window selection strategy. Besides, the random window selection strategy can achieve comparable or even superior results to the SOTA baselines, which also confirms the idea of sliding windows is more suitable than other approaches (e.g., constituent parsing in SelfExplain) for TI tasks. In summary, explanations from ExplainTI are the best compared to prior methods.

Are ExplainTI explanations plausible and trustable for humans? Human evaluation is commonly used to evaluate *plausibility* and *trustability*. We develop the ExplainTI$^+$ system for human evaluation, as illustrated in Figure 4. To this end, 50 human judges annotated a total of 960 samples from WikiTable for both column type prediction and column relation prediction tasks. Judges are graduate students in computer science from various universities. Each judge was provided the evaluation criteria with a corresponding description. The models to be evaluated were anonymized and shuffled. Judges were asked to rate them according to the evaluation criteria alone. Human judges were shown the following: (i) input, (ii) gold label, (iii) predicted label, and (iv) explanations from baselines and ExplainTI-RoBERTa.

Following the previous study [18], we analyze the *plausibility* adopting two criteria: adequate justification and understandability. We evaluate the adequacy of the explanation by asking judges:"Does the explanation adequately justify the model prediction?" The understandability metric evaluates whether a human judge can understand the explanations presented by the model. Figure 5(a) shows that ExplainTI achieves 62% and 43% improvement over the best-performing baseline (i.e., SelfExplain) in terms of adequacy and understandability, respectively. This shows that humans perceived ExplainTI explanations as more plausible compared to the baselines.

To evaluate *trustability*, we follow the same experimental setup as [18], [42] to compute the mean trust score. The trust score ranges from 1-5, reflecting how much human judges trust the explanation. Figure 5(b) depicts the mean trust score of ExplainTI in comparison to the baselines. It is observed that our multi-view explanations are perceived more trustworthy for humans.

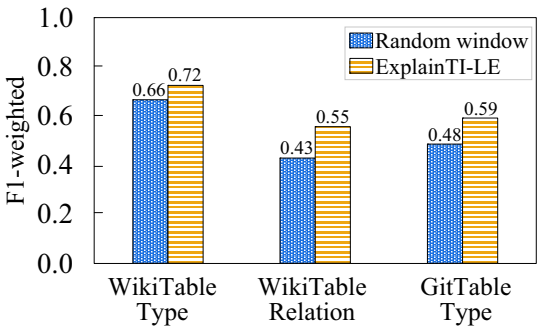


Fig. 3. *Sufficiency* analysis.

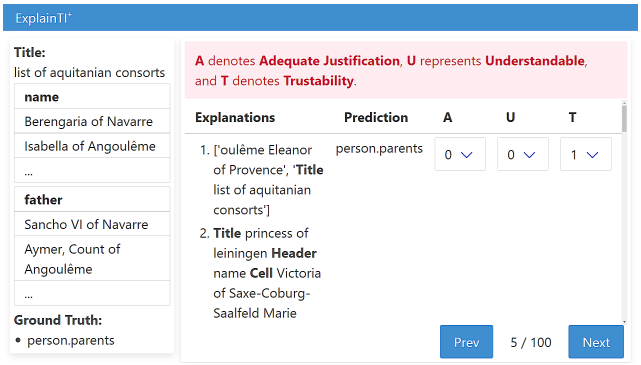
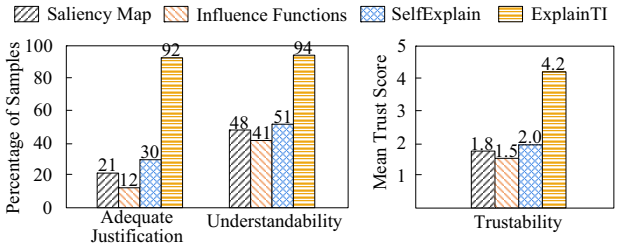


Fig. 4. The illustration of ExplainTI$^+$ system.

Case study of explainability. We visualize a case of explainable column type prediction on WikiTable, the results are depicted in Figure 6. Figure 6 plots: (i) the input column and its prediction label, (ii) the local explanations (relevant windows), (iii) the global explanations (similar samples), and (iv) the structural explanations (influential neighbors).

The prediction of the column type is *location.country* and *location.location*. Our proposed ExplainTI produces multi-view explanations: (i) From the local view, "Rica Guatemala Kenya" is the most relevant window phrase to the input sample and prediction. (ii) From the global view, the most similar sample has the same label as the prediction. (iii) From the structural view, the most influential neighbor has also the same label, which can increase human's trust in the predicted results. According to the above explanations, it is easy and quick to verify the prediction is correct. We can observe that the explanations are (i) relevant to the current input and predictions and (ii) understandable to humans. In other words, the explanations are *good*. In summary, multi-view explanations produced by ExplainTI are meaningful and faithful, which can speed up manual verification.

Online simulation. We conduct an experimental comparison on predicting with/without explanations to verify that our model can genuinely improve the work efficiency of the human experts in evaluating the labels. The experiment is conducted on WikiTable and each model makes predictions on 30 samples that are randomly partitioned. To simulate the real scenario, the derived results by both models are presented to human experts who verify the correctness of the prediction. Three human experts from Huawei Cloud verify the correctness of the prediction simultaneously. An additional



(a) Plausibility analysis (b) Trustability analysis

Fig. 5. *Plausibility* and *trustability* analysis.

expert is responsible for observing the validation in real-time, and will consult with the three experts when the results are inconsistent to obtain a unified decision. The results show that the human experts expend around **19% less time** evaluating the results by our model than those by the one without explanations, which confirms that the explanations provided by our model are beneficial for the human verification process.

D. Ablation Study (RQ3)

Next, we analyze the effectiveness of each proposed module of ExplainTI (i.e., LE, GE, and SE) by comparing ExplainTI with its variants without the key module. Besides, we develop a pre-processing (PP) step to choose unduplicated values of cells. The results are presented in Table III.

ExplainTI vs. ExplainTI w/o LE. We can observe that LE can bring a slight performance boost in most cases. This is expected as LE focuses on providing explanations from a local view. As mentioned in Section IV-C, its role is mainly reflected in explainability.

ExplainTI vs. ExplainTI w/o GE. Like LE, GE is a module dedicated to producing explanations from a global view. GE has little effect on the performance, while GE is sufficient and vital in multi-view explanations, shown in Section IV-C.

ExplainTI vs. ExplainTI w/o SE. We can observe that the F1 drops 1.24% on WikiTable on average. This confirms that aggregating contextual information based on SE greatly helps to train effective TI models. We also observe that the SE brings relatively low improvement on GitTable. This attributes to the nature of this dataset, as it is relatively much easier for ExplainTI to achieve the extremely high performance, i.e., 0.982 F1-weighted, in this dataset.

ExplainTI vs. ExplainTI w PP. We analyze the influence of the same values in cells. We can find that the performance is almost the same using PP. In other words, choosing unduplicated values can not bring noticeable performance improvement.

E. Efficiency Analysis (RQ4)

The goal of ExplainTI is to speed up the manual verification process. However, there is a trade-off between the explainability and efficiency. In other words, providing explanations entails extra computation. We analyze the efficiency of ExplainTI in this section. To be more specific, we conduct an efficiency analysis to evaluate the training time and test time brought by each explainable module (i.e., LE, GE, and SE).

The results are reported in Table V. For the training time, LE and SE hardly increase the training time since they are

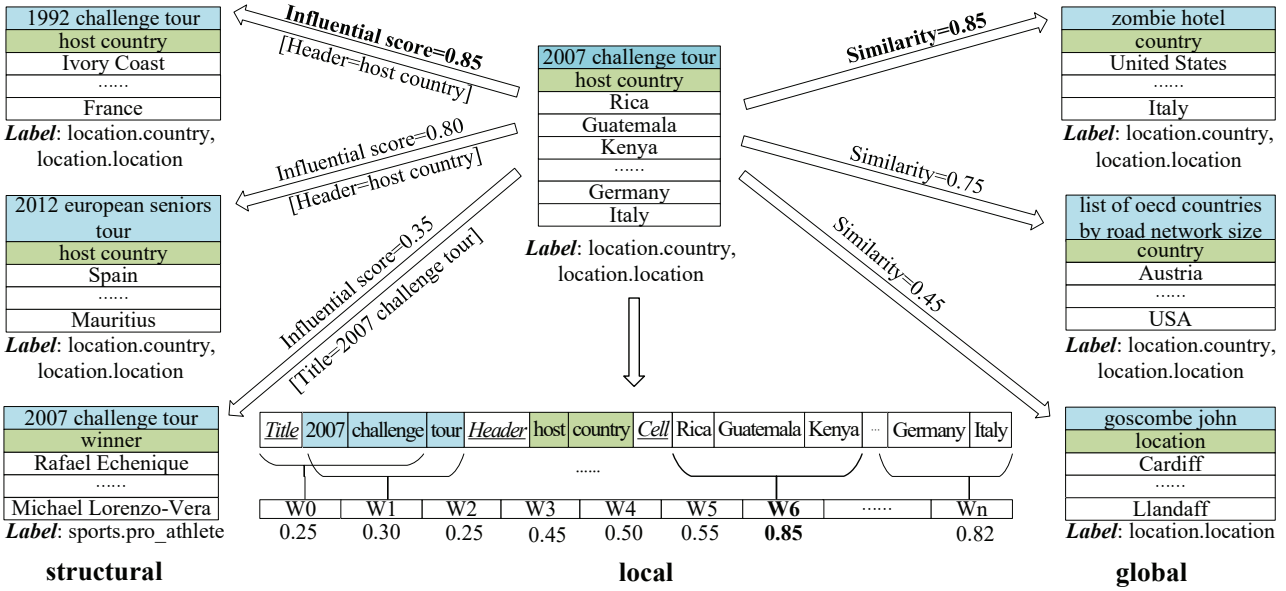


Fig. 6. A case study of explainability on WikiTable. We visualize a case of explainable column type prediction. The local explanations denote the relevant windows, the global explanations represent the similar samples in the training set, and the structural explanations denote the influential neighbors to the input.

TABLE V
EFFICIENCY ANALYSIS. BASE DENOTES EXPLAINTI WITHOUT ANY EXPLAINABLE MODULES, BASE+LE REPRESENTS BASE WITH SOLELY LE, BASE+GE DENOTES BASE WITH SOLELY GE, AND BASE+SE REPRESENTS BASE WITH SOLELY SE. "S" DENOTES SECONDS AND "M" DENOTES MINUTES.

Method	Wiki-Type		Wiki-Relation		Git-Type	
	train	test	train	test	train	test
Base	354.2m	9.5s	54.2m	1.1s	17.5m	1.3s
Base+LE	359.7m	13.1s	57.9m	2.8s	18.1m	2.6s
Base+GE	576.7m	13.4s	76.7m	1.6s	30.8m	2.6s
Base+SE	353.9m	16.3s	53.9m	2.1s	18.1m	3.8s
ExplainTI	581.6m	31.4s	79.9m	5.4s	31.1m	8.9s

computed in parallel on the GPU. We can also observe that GE requires relatively more training time. This is because the retrieval process is computed on the CPU, which can be more efficient using GPU implementation, and we leave it in our further work. For the test time, each explainable module incurs result in more test time. However, it is acceptable because the increased test time (i.e., only several seconds) is negligible compared to the training time. Overall, ExplainTI makes a significant trade-off between the explainability and efficiency.

F. Sensitivity Analysis (RQ5)

We further justify the sensitivity of the proposed ExplainTI by conducting the following four sets of experiments.

Sensitivity to α and β. We conduct a sensitivity analysis of the loss weights α and β used in Eq. 11, and change these two parameters at the same time. Figures 7(a) and 7(b) show that F1-weighted are stable across different settings of loss weights, indicating that ExplainTI works well under different α and β preferences.

Sensitivity to the sampling size r. We perform a sensitivity analysis of the sampling size r in Eq. 5 and Eq. 6. The result are depicted in Figures 7(c) and 7(d). The performance first increases and then slightly drops when r grows, as larger r with more neighbors leads to over-smoothing and intractable

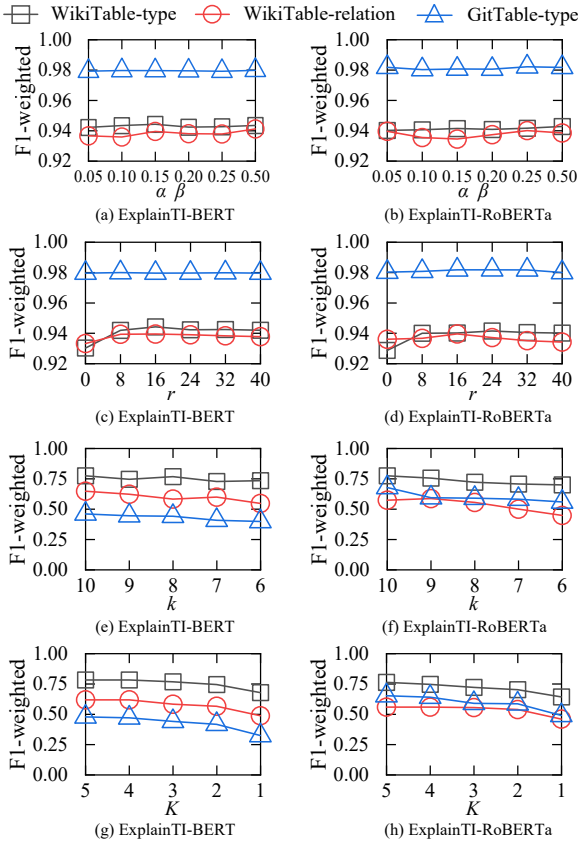


Fig. 7. *Sensitivity* analysis.

learning. We choose $r = 16$ in our experiments due to the best performance.

Influence of the window size k. We conduct a sensitivity analysis of the windows size k used in LE. Figures 7(e) and 7(f) show the predictive performance of ExplainTI-LE under different k. We observe that the F1-weighted drops slowly when k decreases, which is relatively stable. This demonstrates

that LE is robust to the windows size k.

Influence of K. Finally, we further evaluate how different K influence the predictive performance of ExplainTI-LE in Section IV-C. In Figures 7(g) and 7(h), the F1-weighted drops slowly when K decreases. We find that the performance drop is within an acceptable range, i.e., even if using top-1 local explanation, our method still achieves comparable even better performance than baselines using top-3 or top-10 explanations. As mentioned before, ExplainTI-GE and ExplainTI-SE can achieve considerable performance only using top-1 explanation (i.e., the most influential sample), which is shown in Table IV. In other words, changing K will not influence the predictive performance of ExplainTI-GE and ExplainTI-SE. The benefit is that our method does not require enormous explanations to reflect predicted labels, which saves much effort in manual verification.

V. Related Work

A. Table Interpretation

Table interpretation aims at inferring meta-information about tables, such as column types and relations between columns, playing an essential role in data management. Existing column type prediction models enjoy the recent advances in machine learning. Sherlock [37] applies neural networks on multiple feature sets extracted from individual column values. Based on Sherlock, Sato [10] incorporates additional topic modeling components. Other models such as ColNet [43], HNN [20] use external knowledge bases on top of machine learning models to improve column type prediction. EXACTA [5] explains column type prediction using knowledge graph reasoning. There is another line of work that focuses on relation extraction. Harmouch et al. [44] discover meaningful headers via leveraging existing table headers from web tables.

Researchers also try to represent tabular data by deep language models. TaBERT [9] is a pre-trained encoder for joint understanding of textual and tabular data. TURL [7] proposes a pre-training/fine-tuning framework for table understanding tasks. Doduo [6] is a unified table interpretation framework based on pre-trained language models that multi-task learning. These studies based on pre-trained language models can only capture related information in the same table but ignore tabular structure. TCN [8] considers both intra-table and inter-table information by aggregating various types of implicit connections between tables of the same cell value or position. In contrast, we aggregate contextual information at the column level. Recent tabular data representation learning methods have achieved considerable performance. However, they fail to provide any explanations to explain the prediction.

B. Explainable Neural Text Classifiers

Prior work in explainability for neural text classification predominantly follows two lines. One line of explainable neural text classification methods is *post-hoc*, which explains predictions for previously trained models based on model internals. Gradient [38], [45] and relevance [42] based methods produce explainable aspects locally. Although the above methods focus on local explainability, studies such as [19] aim to retrieve influential training samples for global explainability. Another line of work is *inherently explainable* whose explainability is built-in and optimized jointly with the end task. While post-hoc methods are often the only option for already-trained models, inherently explainable models may provide greater transparency since explanation capability is embedded directly within the model. Attention-based methods [15], [46]–[48] are one of the commonly used explainability tools. And the attention mechanism is able to provide faithful explanations [49]–[51]. There are works [39], [52] explore collecting rationales influenced the corresponding predictions. A class of inherently explainable classifiers explain model predictions locally using human-understandable high-level concepts such as prototypes [53] and explainable classes [54]. SelfExplain [18] is similar in spirit to prototypes [53] but additionally provides explanations from global view for the neural text classification task. However, the above methods focus on normal text, having limitations in handling tabular data.

VI. Conclusions and Future Work

In this paper, we study the problem of explainable table interpretation via our presented ExplainTI framework. ExplainTI explains table interpretation using multi-view explanations. Tables are firstly converted to sequences and graphs. Then, a pre-trained transformer encoder is trained to aggregate contextual information and provide multi-view explanations. Extensive experimental results on both Web tables and database tables benchmarks demonstrate the superiority of ExplainTI compared with the state-of-the-art methods. We also systematically analyze the explainability of our framework in terms of sufficiency, plausibility, and trustability.

Based on our analysis, the main limitations and pitfalls of our work can be broken down into the following two cases: (i) ExplainTI is able to provide explanations without manual intervention. Nevertheless, ML models always have an error rate, e.g., providing incorrect predictions or explanations. Manual calibration is necessary to solve the errors, which is a post-processing steps for all ML models. (ii) We focus on explaining the predictions via data instead of the explainability from pre-trained models, and we employ pre-trained models to extract features. In other words, our explainability will not rely on the explainability within the models. Using pre-trained big models does not decrease our explainability. Therefore, We overlook the explainability within models since improving the explainability of these models is outside the scope of our work.

In the future, it is of interest to explore the explainability of LMs and study other explainable data management tasks.

Acknowledgment

This work was supported in part by the NSFC under Grants No. (62025206, 61972338, and 62102351) and Ningbo Science and Technology Special Projects of China under Grant No. 2021Z095. Lu Chen is the corresponding author of the work.

REFERENCES

[1] D. studio, 2022. [Online]. Available: https://datastudio.google.com/
[2] P. BI, 2022. [Online]. Available: https://powerbi.microsoft.com/
[3] Tableau, 2022. [Online]. Available: https://www.tableau.com/
[4] Trifacta, 2022. [Online]. Available: https://www.trifacta.com/
[5] Y. Xian, H. Zhao, T. Y. Lee, S. Kim, R. Rossi, Z. Fu, G. De Melo, and S. Muthukrishnan, "Exacta: Explainable column annotation," in *SIGKDD*, 2021, pp. 3775–3785.
[6] Y. Suhara, J. Li, Y. Li, D. Zhang, Ç. Demiralp, C. Chen, and W.-C. Tan, "Annotating columns with pre-trained language models," in *SIGMOD*, 2022, pp. 1493–1503.
[7] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu, "Turl: table understanding through representation learning," in *VLDB*, 2020, pp. 307–319.
[8] D. Wang, P. Shiralkar, C. Lockard, B. Huang, X. L. Dong, and M. Jiang, "Tcn: Table convolutional network for web table interpretation," in *WWW*, 2021, pp. 4020–4032.
[9] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "Tabert: Pretraining for joint understanding of textual and tabular data," in *ACL*, 2020, pp. 8413–8426.
[10] D. Zhang, M. Hulsebos, Y. Suhara, Ç. Demiralp, J. Li, and W.-C. Tan, "Sato: contextual semantic type detection in tables," in *VLDB*, 2020, pp. 1835–1848.
[11] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang, "Data cleaning: Overview and emerging challenges," in *SIGMOD*, 2016, pp. 2201–2206.
[12] L. Chen, D. Lee, and T. Milo, "Data-driven crowdsourcing: Management, mining, and applications," in *ICDE*, 2015, pp. 1527–1529.
[13] J. Fan, G. Li, B. C. Ooi, K.-l. Tan, and J. Feng, "icrowd: An adaptive crowdsourcing framework," in *SIGMOD*, 2015, pp. 1015–1030.
[14] L. Xu and X. Zhou, "A crowd-powered task generation method for study of struggling search," *DSE*, vol. 6, no. 4, pp. 472–484, 2021.
[15] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015, pp. 2048–2057.
[16] S. Serrano and N. A. Smith, "Is attention interpretable?" in *ACL*, 2019.
[17] S. Jain and B. C. Wallace, "Attention is not explanation," in *NAACL*, 2019.
[18] D. Rajagopal, V. Balachandran, E. Hovy, and Y. Tsvetkov, "Selfexplain: A self-explaining architecture for neural text classifiers," in *EMNLP*, 2021.
[19] X. Han, B. C. Wallace, and Y. Tsvetkov, "Explaining black box predictions and unveiling data artifacts through influence functions," in *ACL*, 2020, pp. 5553–5563.
[20] J. Chen, I. Horrocks, E. Jimenez-Ruiz, and C. Sutton, "Learning semantic annotations for tabular data," in *IJCAI*, 2019, pp. 2088–2094.
[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019.
[22] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint*, 2019.
[23] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *ICML*, 2017, pp. 3145–3153.
[24] Y. Hou, J. Zhang, J. Cheng, K. Ma, R. T. B. Ma, H. Chen, and M. Yang, "Measuring and improving the use of graph information in graph neural networks," in *ICLR*, 2020.
[25] F. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *ICLR*, 2020.
[26] Y. A. Malkov and D. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *TPAMI*, vol. 42, no. 04, pp. 824–836, 2020.
[27] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, 2019.
[28] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *AAAI*, 2019.
[29] Y. Lin, Y. Meng, X. Sun, Q. Han, K. Kuang, J. Li, and F. Wu, "Bertgcn: Transductive text classification by combining gnn and bert," in *ACL FINDINGS*, 2021.
[30] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NeurIPS*, 2017, pp. 1025–1035.
[31] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.
[32] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *ICLR*, 2017.
[33] R. Ghaeini, X. Z. Fern, and P. Tadepalli, "Interpreting recurrent and attention-based neural models: a case study on natural language inference," in *EMNLP*, 2018.
[34] M. Hulsebos, Ç. Demiralp, and P. Groth, "Gittables: A large-scale corpus of relational tables," *arXiv preprint*, 2021.
[35] R. V. Guha, D. Brickley, and S. Macbeth, "Schema. org: evolution of structured data on the web," *Communications of the ACM*, pp. 44–51, 2016.
[36] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *ISWC*, 2007, pp. 722–735.
[37] M. Hulsebos, K. Hu, M. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, Ç. Demiralp, and C. Hidalgo, "Sherlock: A deep learning approach to semantic data type detection," in *SIGKDD*, 2019, pp. 1500–1508.
[38] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," in *ICLR*, 2014.
[39] J. DeYoung, S. Jain, N. F. Rajani, E. Lehman, C. Xiong, R. Socher, and B. C. Wallace, "Eraser: A benchmark to evaluate rationalized nlp models," in *ACL*, 2020, pp. 4443–4458.
[40] A. Jacovi and Y. Goldberg, "Towards faithfully interpretable nlp systems: How should we define and evaluate faithfulness?" in *ACL*, 2020.
[41] S. Jain, S. Wiegreffe, Y. Pinter, and B. C. Wallace, "Learning to faithfully rationalize by construction," in *ACL*, 2021, pp. 4459–4473.
[42] C. Singh, W. J. Murdoch, and B. Yu, "Hierarchical interpretations for neural network predictions," in *ICLR*, 2018.
[43] J. Chen, E. Jiménez-Ruiz, I. Horrocks, and C. Sutton, "Colnet: Embedding the semantics of web tables for column type prediction," in *AAAI*, 2019, pp. 29–36.
[44] H. Harmouch, T. Papenbrock, and F. Naumann, "Relational header discovery using similarity search in a table corpus," in *ICDE*, 2021, pp. 444–455.
[45] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *ICML*, 2017, pp. 3319–3328.
[46] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *ICLR*, 2015.
[47] M. Zhu, D. Shen, L. Xu, and X. Wang, "Scalable multi-grained cross-modal similarity query with interpretability," *DSE*, vol. 6, no. 3, pp. 280–293, 2021.
[48] Y. Lyu, H. Yin, J. Liu, M. Liu, H. Liu, and S. Deng, "Reliable recommendation with review-level explanations," in *ICDE*, 2021, pp. 1548–1558.
[49] S. Vashishth, S. Upadhyay, G. S. Tomar, and M. Faruqui, "Attention interpretability across nlp tasks," *arXiv preprint arXiv:1909.11218*, 2019.
[50] A. Galassi, M. Lippi, and P. Torroni, "Attention in natural language processing," *TNNLS*, vol. 32, no. 10, pp. 4291–4308, 2020.
[51] A. K. Mohankumar, P. Nema, S. Narasimhan, M. M. Khapra, B. V. Srinivasan, and B. Ravindran, "Towards transparent and explainable attention models," in *ACL*, 2020, pp. 4206–4216.
[52] D. Croce, D. Rossini, and R. Basili, "Auditing deep learning processes through kernel-based explanatory models," in *EMNLP*, 2019, pp. 4037–4046.
[53] D. Alvarez-Melis and T. S. Jaakkola, "Towards robust interpretability with self-explaining neural networks," in *NeurIPS*, 2018, pp. 7786–7795.
[54] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang, "Concept bottleneck models," in *ICML*, 2020, pp. 5338–5348.