DOI Link: https://doi.org/10.61586/xpVdz
Vol.70, Issue.10, Part.1, Oct 2025, PP.29-50

Vehicle Data Asset Recognition Based on Pattern Matching

Jiaming Chen[1,2], Changyun Huang[2], Xiangyang Wang[1], Chengyu Tan[1], Wei Luo[1] and Zhenxing Dong[2,*]

[1] State Key Laboratory of Intelligent Vehicle Safety Technology, Chongqing Changan Automobile Co.,Ltd., Chongqing 401133, China
[2] School of Cyberspace Security and Information Law, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

Received Feb 2025 Accepted Sep 2025 Published Oct 2025

ABSTRACT

Existing data asset identification schemes predomi-nantly rely on manual categorization methods, which are often associated with high costs and low accuracy. Consequently, there is an urgent need for more intelligent and automated solutions to enhance the efficiency and precision of data asset identification. To address this challenge, this paper proposes a pattern-matching-based vehicle data asset identification scheme, which leverages a pre-trained model to automatically generate asset labels for data fields, thereby enabling more efficient classification and identification of vehicle data assets. This model scheme generates corresponding classification prediction results based on the sequence of each input data item in the tabular data and assigns appropriate asset labels to each column of data. The experimental results show that, compared to other schemes, the scheme proposed in this paper demonstrates certain advantages in both recognition efficiency and accuracy.

KEYWORDS

Internet of Vehicles (IoV), Data Asset Identification, Pattern Matching.

I. INTRODUCTION

From the perspective of generating value from data, data itself does not directly generate benefits. Only through the analysis and development of the information contained within the data—by processes such as data organization, processing, and classification—can raw data be transformed into intuitive and valuable data assets[1]. With the continuous advancement and widespread adoption of technology, the surge in automobiles and related devices has led to an explosive growth in data generation, characterized by the richness of data types and the complexity of structures. This undoubtedly exacerbates the challenges in identifying and utilizing data assets. Data asset identification is key in data management and risk analysis. It depends on effective tools to recognize, classify, and manage digital assets.As a core issue in data management and risk analysis, data asset identification relies on effective technical means to assist in recognizing, classifying, and managing digital assets. In this context, the development of artificial intelligence plays a pivotal role in digital asset identification.

Classifying and identifying data assets often presents challenges. One common issue is the presence of diverse labels for the same type of data within a database. For instance, date data might be labeled as "date", "time", or other variations. Traditionally, manual auditing methods struggle to cope with such complexity and diversity, resulting in high costs and limited accuracy. Therefore, how to efficiently integrate information from data sources and generate complete business semantic information for data fields through more intelligent and automated methods, thereby assigning unified asset labels to these same types of data entities and achieving more efficient enterprise data asset identification, is an urgent research issue for realizing advanced informatization and business digitization.

This paper addresses the issues in existing vehicle data asset identification processes, which heavily rely on manual adjustments and configurations, as well as the diversity of data labels, leading to high costs and low accuracy. This paper proposes a pattern-matching-based vehicle data asset identification scheme. By analyzing data field entities input into the identification model, the scheme assigns unified asset labels to different types of data entities, thereby achieving the classification and identification of data assets. This scheme improves the efficiency of enterprise data asset identification and significantly reduces the costs associated with traditional manual methods. Furthermore, the proposed approach integrates prior knowledge from the Internet of Vehicles (IoV) domain into the model's classification and identification process, enhancing the model's ability to parse and distinguish different types of vehicle data and improving its overall performance. Ultimately, this enables precise identification and efficient management of vehicle data assets. The main contributions of this paper are as follows:

Traditional vehicle data asset identification suffers from high costs and low accuracy due to heavy reliance on manual adjustments. To overcome this, this paper proposes a prior feature embedding-based vehicle data asset identification model.The model distinguishes different types of data field entities and generates corresponding asset labels, enabling intelligent classification and identification of data assets.

To improve the low accuracy of existing pre-trained language models in vehicle data asset identification, this paper proposes a BERT-MLP-based prior feature embedding module. The module enriches input-level information by embedding tabular sequence data with BERT and combining it with MLP-extracted prior features. This integration significantly improves vehicle data asset identification accuracy.

The final experimental results demonstrate that, compared to existing schemes, the scheme proposed in this paper not only improves the efficiency of the vehicle data asset identification process but also enhances the accuracy of vehicle data identification.

II. Related Work

Current studies on data asset identification predominantly concentrate on manual approaches. These methods encompass defining asset boundaries, normalizing attributes, performing comprehensive inventories of existing data assets, and implementing automated registration for newly added assets to ensure precise identification. However, most current asset identification schemes still suffer from high costs and low accuracy.

In the field of data asset identification, although some automated tools have been developed, their application is largely confined to the data collection phase. Current mainstream solutions (e.g., nmap[2], real-time traffic analysis tools[3],[4] can achieve basic network asset discovery through active scanning and traffic capture but struggle to effectively parse the metadata characteristics of data assets. For instance, raw data collected by systems often lacks deep recognition of field meanings, business relevance, lifecycle status, and other meta-attributes, still requiring manual analysis and semantic annotation of data structures[5]. Current studies also highlight several persistent limitations. These include varying valuation criteria, inadequate approaches to data quality and security issues, difficulties in managing large-scale dynamic datasets, and insufficient attention to solution efficiency and scalability. Future research needs to address these shortcomings by exploring more efficient, flexible, and adaptive data asset identification frameworks to better align with rapidly evolving digital environments and ensure the secure and efficient utilization of data assets.

In leveraging pattern matching to effectively analyze and process massive network data, existing methods mainly focus on improving matching efficiency and accuracy using algorithms or deep learning models. However, challenges remain in handling data scale, heterogeneity, and deep data understanding[6]. To address the difficulty of efficiently integrating and analyzing massive heterogeneous data generated by IoT devices, literature[7] proposed a Hierarchical Semantic Matching Algorithm (HSMA) for IoT heterogeneous data. Through three layers of mapping—feature classification matching, relational feature clustering matching, and hybrid element matching—it automatically completes pattern matching for IoT heterogeneous data, establishing correspondences between data of different sources and structures to achieve effective information integration and interoperability. To overcome the limited understanding of data distribution characteristics in matching tasks, literature[8] adopted a novel approach using statistical features of probability distributions. This method enhanced pattern matching performance by extracting distinctive data features and applying BP neural networks for improved efficiency and accuracy. However, this method only focuses on column-level matching and is unsuitable for complex pattern matching. As data scales continue to grow, matching space and frequency increase dramatically, leading to inefficiency. To address the challenges of massive heterogeneous IoT data and subjective attribute naming in databases, literature[9] developed an entity classification-based schema matching approach. This method specifically targets the variability in database design conventions while handling large-scale IoT data diversity. It extracts features based on the semantic information of data instances, classifies features using the Naive Bayes algorithm, and performs corresponding schema matching for sub-schemas. However, given the enormous scale and heterogeneity of IoT data, no matter how standardized the database design may be, the definitions of attribute names and their meanings still vary from person to person. Relying solely on semantic information such as attribute names for fuzzy classification is impractical, as this classification hinges on subjective human understanding and fails to reflect the intrinsic characteristics of the source data.

Existing research on using pretrained language models for deep learning classification and prediction of tabular data primarily focuses on achieving new performance benchmarks in data classification tasks. This effectively supports data management and analysis by enabling automated identification and classification of digital assets. To address the low accuracy of BERT on tabular data, literature[10] proposed a TABERT pre-trained model. This model significantly improves performance in understanding and parsing tasks involving both text and structured data by jointly learning representations of text and tabular data. It demonstrates competitiveness on the SPIDER dataset, providing a robust foundation for handling complex information classification and understanding tasks across text and tables. Literature[11] proposed a multi-task learning framework that annotates table columns using pre-trained language models, advancing the field of data classification. It predicts column types and inter-column relationships, enhancing the understanding of tabular data and achieving new performance heights in data classification tasks, effectively aiding data management and analysis. Literature[12] proposed the TURL framework, which introduces a pre-training/fine-tuning paradigm to relational network tables, learning deep contextual representations to adapt to various table understanding tasks with minimal task-specific fine-tuning. It proposes a structure-aware Transformer encoder and a Masked Entity Recovery objective, demonstrating broad applicability and superior performance across six tasks. Pre-trained models play a core role in data asset analysis, with innovative models such as TABERT, Doduo, TURL, and SEMPROP advancing data classification, understanding, and management. These models significantly improve precision and efficiency by deeply learning representations of text, table, and relational data, providing powerful tools for cross-domain complex information processing. However, these methods and models are trained on general datasets and have limited capabilities in vehicle data identification.

III. Problem Analysis

In a large-scale data system X containing multiple data tables of different asset types, each data table T consists of multiple columns C of varying types, which may represent different entities and attributes. The objective of the data asset identification task is to analyze the data fields in each data table and generate business semantic labels for the table data fields.

Specifically, given a collection of data tables $X = T_1, T_2, \ldots, T_m$, where each data table T_i contains several columns $C_{i1}, C_{i2}, \ldots, C_{in}$, and each data entity v_{ij} in column C_{ij} belongs to the data values of column C_{ij}. The goal of this paper is to predict an appropriate asset type label l_{ij} for each data column C_{ij} in the input data table T_i, where $l_{ij} \in L_{\text{type}}$ and L_{type} represents the set of all label types. Formally, for each column C_{ij} with column values v_{ij}, this paper constructs a classification model $f: V(C_{ij}) \rightarrow L_{\text{type}}$, which generates the asset type label for the column data.

$$\mathrm{f}(\mathrm{v}_{\mathrm{ij}}) = \hat{\mathrm{y}}, \hat{\mathrm{y}} \in \mathrm{L}_{\mathrm{type}} \tag{1}$$

Thus, for each given column value v_{ij}, the model outputs a type label $\hat{y} \in L_{\text{type}}$, representing the predicted asset type label for that column.

IV. Vehicle Data Asset Identification Scheme

A. Overall process

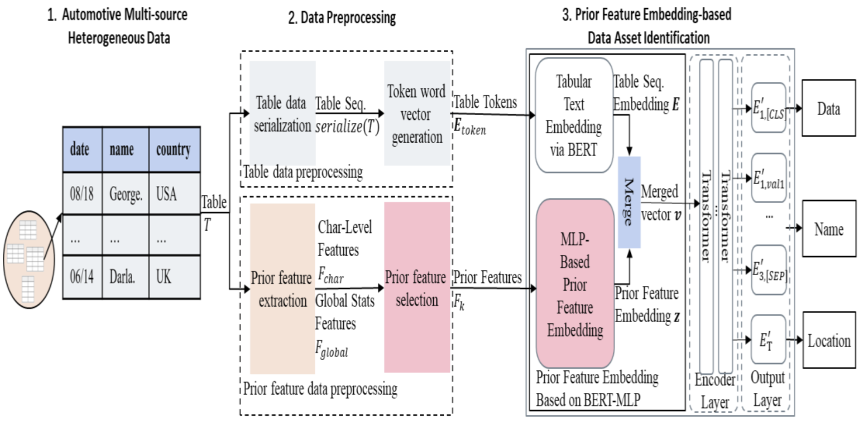

Fig. 1. Data Asset Identification Framework Based on Prior Feature Embedding

This paper proposes a vehicle data asset identification scheme based on prior feature embedding, with the overall framework illustrated in Figure 3-1. This paper aims to achieve intelligent and automated data asset identification and generate business semantic asset labels for tabular data fields. It integrates a pretrained model to extract textual features and employs an MLP to extract prior knowledge features, thereby comprehensively capturing the feature information of the text. The specific workflow is as follows:

First, the table T requiring asset identification is obtained from multi-source heterogeneous vehicle data, and data preprocessing is performed. During the preprocessing phase, the table T is serialized using a multi-item serialization method, converting T into $\mathrm{serialize}(T)$, which is then transformed into word vector inputs E_{token} that the model can process. This allows the entire table T to be input into the model, enabling the model to learn the data structure of the entire table. In the prior feature extraction phase, domain-specific prior knowledge is leveraged to extract key features, including character-level distribution features F_{char} and global statistical features F_{global}. Feature selection is then performed to identify the top k most important features, which are input into the model as prior feature data F_k.

Next, in the BERT-MLP-based prior feature embedding module of the asset identification model, the table sequence embedding E is obtained through the BERT-based table text embedding module, and the prior feature embedding z is obtained through the MLP-based prior feature embedding module. These two embeddings are concatenated to form the combined vector v. Subsequently, a Transformer encoder is used to encode the combined vector v, capturing deeper dependencies and contextual information, and producing a hidden state sequence of the table data. Finally, the output layer of the model generates classification predictions based on the hidden state sequence provided by the encoder. These predictions determine the asset labels for each column of data.

B. Data Preprocessing

1) Table Data Preprocessing

Table Data Serialization. Serializing table data transforms the original tabular text data into a sequence format that BERT can accept, preserving the contextual information between table data. The traditional single-column model concatenates the text of each column individually to form a data sequence, which is then used as the input for the model. Suppose a column C has column values $v_1, v_2, \ldots, v_m$. The serialized sequence is:

$$\mathrm{serialize}_{\text{single}}(C) ::= [\mathrm{CLS}]\ v_1, v_2, \ldots, v_m\ [\mathrm{SEP}] \quad (2)$$

Here, [CLS] and [SEP] are special tokens used to mark the beginning and end of the sequence, respectively. For example, a column of data might be serialized as: [CLS] Happy Feet Cars Flushed Away [SEP]. This serialization transforms the problem into a sequence classification task. Therefore, fine-tuning the BERT model using training data is straightforward, enabling the language model to understand discrete data.

However, the single-column model treats each column or column pair as an independent sequence by serializing them separately. As a result, columns within the same table are processed independently. Consequently, the single-column model typically lacks consideration for the overall structure of the table and fails to leverage the relationships between table columns, thereby capturing insufficient contextual information. Unlike traditional single-column models that use only one [CLS] token in the input, this model employs multi-data-item serialization to insert as many [CLS] tokens as there are columns. This difference leads to a change in the classification algorithm. While the single-column model classifies by predicting a single label for a single sequence, the multi-item serialization predicts as many labels as the number of [CLS] tokens in the input sequence. The learning process begins with the serialization of all tables and lexical analysis in the dataset. By using multi-serialization, the model can leverage the contextual information of the entire table, effectively capturing table context and addressing the issue of the single-column model ignoring inter-column relationships due to independent column processing. For a table $T = (c_i)_{i=1}^{n}$ with n columns, where each column $c_i = \left(v_j^i\right)_{j=1}^{m}$ has m column values, the serialized sequence can be represented as:

$$\begin{aligned} \text{serialize}(T) ::= {} & [\text{CLS}]\, v_1^1, \dots, v_m^1, \\ & [\text{CLS}]\, v_1^2, \dots, v_m^2, \\ & \dots, \\ & [\text{CLS}]\, v_1^n, \dots, v_m^n, \\ & [\text{SEP}] \end{aligned} \tag{3}$$

Token Vector Generation. Token vectors are obtained from each token in the input sequence $\text{serialize}(T)$, representing a vector obtained through a lookup table. For the input sequence $\text{serialize}(T) = [\text{CLS}]\, v_1^1, \dots, v_m^1\, [\text{CLS}]\, v_1^2, \dots$ $[\text{CLS}]\, v_1^n, \dots, v_m^n\, [\text{SEP}]$, each token t_i (i.e., each column value v_j^i) is transformed into a fixed-dimensional vector derived from the pre-trained embedding matrix E_{token}. The token embedding can be represented as:

$$\text{Token embedding} = E_{token}[t_i] \tag{4}$$

For tabular data, the token embeddings of all column values are arranged in sequence to form the input token embeddings E_{token} for the embedding layer:

$$E_{\text{token}} = [E_{\text{token}}(v_1^1), \dots, E_{\text{token}}(v_m^n)] \tag{5}$$

2) Prior Feature Data Preprocessing

Prior Feature Extraction: Prior feature extraction aims to obtain pattern representation information from tabular text data, enhancing the richness of information at the model input level. Specifically, this paper constructs a prior knowledge-based feature space, which includes two types of prior features: global statistical features F_{global} and character-level distribution features

F_{char}. Global statistical features are used to capture column-level distribution patterns, while character-level distribution features are used to build fine-grained data pattern representations.

TABLE I. GLOBAL STATISTICAL FEATURE CATEGORIES AND THEIR MEANINGS

Feature Category	Meaning
Number of Values	Total number of values in the column
Column Entropy	Measures the uniformity of value distribution (higher entropy indicates more uniform distribution)
Proportion of Unique Values	Ratio of unique values to total values
Proportion of Numeric Values	Ratio of values containing numbers
Proportion of Alphabetic Values	Ratio of values containing letters
Mean and Standard Deviation of Numeric Characters	Distribution of the number of numeric characters in values
Mean and Standard Deviation of Alphabetic Characters	Distribution of the number of alphabetic characters in values
Mean and Standard Deviation of Special Characters	Distribution of the number of special characters (e.g., punctuation) in values
Mean and Standard Deviation of Word Count	Distribution of the number of words per value
Null Value Statistics	Percentage, count, and whether the column contains only null or boolean values
Value Length Statistics	Sum, minimum, maximum, median, mode, kurtosis, skewness, and whether all values exist

● Global Statistical Features: Global statistical features provide important information about trends and dispersion in the dataset. Specific features are listed in Table I, which describes high-level statistical characteristics of the columns. For example, the "Column Entropy" feature describes how uniformly numerical values are distributed. Such features help distinguish types containing more repeated values (e.g., gender) from those containing many unique values (e.g., names). Other types, such as weight and sales, may consist of many numeric characters, which are captured by the "Average Number of Numeric Characters in Values."

● Character-Level Distribution Features: Character-level distribution features provide characteristics at the character level, describing the distribution of characters within a column. During the extraction of character-level distribution features, a target character set (96 printable ASCII characters) is constructed, and the frequency of each character in the data column is traversed. For each character, 10 statistical measures are calculated across all samples: existence (whether it appears at least once), universality (whether it appears in all samples), mean, variance, minimum, maximum, median, sum, kurtosis, and skewness. For characters that do not appear, default statistical values are automatically filled to maintain consistent feature dimensions. This results in a total of $96 \times 10 = 960$ features.

Prior Feature Selection: Prior feature selection aims to mitigate the curse of dimensionality caused by high-dimensional features and reduce the model's memory consumption. It measures feature importance based on the total reduction in the Gini impurity criterion brought by these features to a decision tree model. The top k most important features are selected from the global statistical and character-level distribution sets and input into the model. The prior feature selection algorithm is as follows:

Algorithm 1 Prior Feature Selection Algorithm

Require: $N, D, F_{\text{global}}, F_{\text{char}}, L$
Ensure: F_k

```
// Initialize feature lists
// Global statistical feature list
F_global ← [columnentropy, …]
// Character distribution feature list
F_char ←["any", "in", "var", "min", "max", "median", "sum", "kurtosis", "skew"]
// Build decision tree model
Model ← DecisionTreeClassifier(criterion='gini')
X ← pd.DataFrame
Model.fit(X, L)
I ← Model.feature_importances_  // Calculate based on Gini impurity criterion
(I, F) ← sort(I, F)
F_k ← F[0: k]
F_k
return F_k
```

In the prior feature selection algorithm, N represents the number of samples in the dataset, D represents the total number of features in the dataset, F_{global} is the list of global statistical features, F_{char} is the list of character-level distribution features, L is the set of target variable labels, and F_k is the list of the top k most important features.

C. Data Asset Identification Based on Prior Feature Embedding

In the vehicle data asset identification scheme proposed in this paper, inspired by the technique of label embedding fusion[13] and the multi-granularity feature system proposed in a paper[14], a BERT-MLP-based prior feature embedding module is introduced. This module extracts and transforms continuous prior feature data to obtain prior feature embeddings. These embeddings are then concatenated with the table text embeddings generated by the BERT text embedding module to form an enhanced input. This enhances the model's ability to understand and predict text information and prior features, aiming to improve prediction performance for automotive data by enriching the information at the model input level.

1) BERT-MLP-Based Prior Feature Embedding:

The primary function of the BERT-MLP-based prior feature embedding module is to leverage table text features and extracted prior features to enhance the richness of information at the model input level, thereby improving prediction performance for automotive data. In the proposed vehicle data asset identification scheme, the BERT-MLP-based prior feature embedding module is responsible for two main tasks: (1) processing table sequence data through the BERT embedding module, and (2) extracting and transforming continuous prior feature data through the MLP-based prior feature embedding module. The outputs of these two components are concatenated and provided as an enhanced input to the subsequent Transformer encoding layer, enhancing the model's ability to understand and predict text information and prior features. This design not only helps capture complex relationships in the data but also improves the overall performance of the model in vehicle data asset identification. Specific implementation details will be elaborated in later sections.

2) Encoder Layer

The model encoder layer generates context-aware representations that integrate text information and prior feature information. These representations contain rich semantic information extracted from the original input data and additional prior features. The encoding layer essentially consists of a series of Transformer encoding blocks, each containing a multi-head self-attention mechanism and a feedforward neural network. Each Transformer block aggregates contextual information from all column values in the same table (including the virtual [CLS] tokens and itself). The first Transformer layer computes attention vectors by aggregating embeddings of other tokens and prior features based on their similarity to the second [CLS] token. Therefore, for the same token, its attention vector may vary depending on the context in which it appears.

After encoding the data into sequence information embeddings, the Transformer layer transforms the sequence information embeddings into key (K), query (Q), and value (V) embeddings. The context embeddings of the sequence are computed as the weighted average of the value embeddings of all column embeddings, where the weights are calculated based on the similarity between the query embeddings and the key embeddings. By separating the key embeddings and query embeddings, the model can compute context embeddings in an asymmetric manner. Additionally, Transformer-based models typically have multiple attention heads (e.g., 12 attention heads at time t). Different attention heads have different parameters for computing K , Q , and V , enabling them to capture different features of the input data as a whole. Finally, the output of a Transformer block is transformed into the same dimension size as the input (e.g., 768 for BERT), allowing the output of the previous Transformer block to be directly used as the input of the next Transformer block. Since the model inserts virtual [CLS] tokens for each column, these tokens can be viewed as contextualized column representations. The encoding layer takes the input v from the embedding layer and passes it to the self-attention mechanism, where v serves as the query (Q), key (K), and value (V).

$$A = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6}$$

where Q , K , and V are matrices obtained from v through different linear transformations, and d_k is the dimension of the key vectors.

For multi-head self-attention, the above process is independently performed multiple times in different projection spaces. The results are then concatenated and transformed through a linear layer to obtain the final attention output. Subsequently, this attention output is passed through a feedforward neural network (FFN), which consists of two linear layers with a GELU activation function in between. The computation of the FFN can be expressed as:

$$\text{FFN}(A) = W_2\, \text{GELU}(W_1 A + b_1) + b_2 \tag{7}$$

Finally, this output is passed through layer normalization and a residual connection, and then forwarded to the next encoding layer or output layer. Through the stacked structure of multiple Transformer blocks, the model can gradually extract semantic features from local to global levels, providing a robust contextual representation foundation for downstream tasks. Ultimately, the output of the final layer is a set of hidden state vectors that have undergone deep contextual modeling. These vectors integrate global semantic information while retaining the fine-grained features of the input sequence. They provide a high-quality representation foundation for subsequent tasks.

3) Output Layer

For the output layer, its task is to make the final prediction based on the contextual hidden state vectors generated by the encoding layer. After processing through multiple Transformer encoders in the encoding layer, the model generates context-aware representations that integrate text information and prior feature information. These representations contain rich semantic information extracted from the original input data and additional prior features. The process of generating predictions in the output layer can be simplified into the following formulaic representation:

$$\hat{y} = \text{Output Layer}(W \cdot h + b) \tag{8}$$

where h represents the hidden state of the final layer output by the encoding layer, which integrates information from all previous layers, including BERT word vectors and discrete feature vectors generated by the MLP; W and b are the weight matrix and bias term of the linear transformation, respectively, used to map the encoded high-dimensional vector into the category space to obtain scores or probability distributions for each category; and $\hat{y}$ is the final model prediction result for each data item sequence.

Ultimately, the output layer of the model provides the classification prediction result $\hat{y}$ for each input table data sequence, assigning the most likely type label to each column of data. This prediction result not only considers the content of multiple table data sequences but also incorporates the prior features of the entire table data, achieving more accurate data asset identification.

V. Prior Feature Embedding Based on BERT-MLP

The BERT-MLP-based prior feature embedding is designed to leverage table text features and table prior features to enhance the richness of information at the model input level. This approach improves prediction performance for automotive data. In the specific implementation process, BERT can effectively handle discrete text data. However, directly inputting extracted prior feature data (continuous numerical data) as strings may disrupt their statistical properties and exceed the BERT sequence length limit. Therefore, this paper proposes a BERT-MLP-based prior feature embedding module, as shown in Figure 2, which uses BERT to obtain table sequence embeddings E and MLP to obtain prior feature embeddings z. Finally, the table sequence embeddings E and prior feature embeddings z are concatenated and input into the subsequent Transformer encoding layer. This constructs an enhanced input to improve the model's understanding and prediction capabilities.

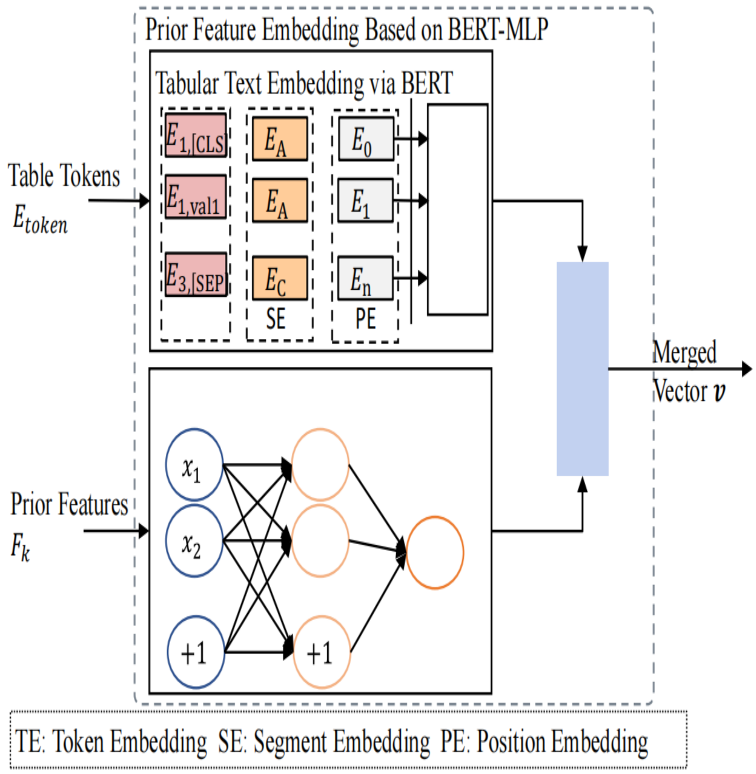

Fig. 2. Prior Feature Embedding Based on BERT-MLP

A. BERT-Based Table Text Embedding

The primary role of BERT-based table text embedding is to help the model distinguish between different sequences of input data. When processing text data, the embedding layer of the BERT pre-trained model obtains the input token vectors E_{token} as token embeddings. It also calculates segment embeddings and position embeddings based on these input token vectors. Once these three embeddings are obtained, they are summed to form the final input vector, which is then passed to the model's encoding layer for further processing.

Token Embedding: Token embedding maps each sub-word in the sequence to a fixed-dimensional vector through a lookup table. In this paper, token embeddings are obtained through the processes of table data serialization and token vector generation.

Segment Embedding: Segment embeddings are used to distinguish between different segments in the input token vectors. When processing tables, each column can be considered a segment, so all values in a column are assigned the same segment label. For a table with n

columns, each token t_i in the segment embedding matrix E_{segment} is assigned a segment label based on its column number. If token t_i belongs to the k-th column, its segment embedding is:

$$\text{Segment Embedding}(\text{t}_\text{i}) = \text{E}_{\text{segment}}(\text{k}) \tag{9}$$

For example, all tokens in the first column will have the segment embedding $E_{\text{segment}}(1)$, all tokens in the second column will have $E_{\text{segment}}(2)$, and so on. The segment embedding matrix is:

$$\begin{aligned} E_{\text{segment}} = [&E_{\text{segment}}(1), E_{\text{segment}}(1), \ldots, \\ &E_{\text{segment}}(2), E_{\text{segment}}(2), \ldots, \\ &\ldots, \\ &E_{\text{segment}}(n)] \end{aligned} \tag{10}$$

Position Embedding: Position embeddings are used to mark the position of each token vector in the input sequence. Since a table is a two-dimensional structure, position embeddings need to encode the position of each token in the sequence. If token t_i is located at the i-th position (starting from 1), its position embedding is:

$$\text{Position Embedding}(t_i) = E_{\text{position}}(i) \tag{11}$$

For table data, the position embedding matrix E_{position} is:

$$E_{\text{position}} = \left[E_{\text{position}}(1), E_{\text{position}}(2), \ldots, E_{\text{position}}(n \times m)\right] \tag{12}$$

Here, $n \times m$ is the length of the entire input sequence. The final input embedding representation is the sum of token embeddings, segment embeddings, and position embeddings. For each token, its total embedding representation is:

$$\text{Input Embedding}(t_i) = E(t_i) = \begin{cases} E_{\text{token}}(t_i) \\ +E_{\text{segment}}(k) \\ +E_{\text{position}}(i) \end{cases} \tag{13}$$

Therefore, the final embedding matrix for the entire text input sequence is:

$$E = [E(t_1), E(t_2), \ldots, E(t_{n \times m})] \tag{14}$$

B. Prior Feature Embedding Based on MLP

The MLP-based prior feature embedding utilizes MLP to map the extracted continuous prior features into a high-dimensional space, enhancing feature representation capabilities. MLP is a feedforward neural network model that primarily uses multiple hidden layers and nonlinear activation functions to map input data to output, thereby modeling complex nonlinear relationships. The proposed asset identification model incorporates a three-layer MLP module in the embedding layer, consisting of two fully connected layers and a Dropout layer. This module serves as a universal function approximator, mapping continuous prior feature data F_k into a high-dimensional space and transforming it into discrete prior feature vectors z.

This process can be mathematically represented as follows:

Assume there is a prior feature $x_{\text{feature}} \in F_k$, where k represents the number of features. The MLP module can be represented as a function $f_{\text{MLP}}(\cdot)$, which takes x_{feature} as input and outputs a discrete feature vector z:

$$z = f_{\text{MLP}}(x_{\text{feature}}; \theta) \tag{15}$$

where θ is the set of MLP parameters. The MLP module typically contains one or more hidden layers, each of which can be represented as:

$$\begin{gathered} h_1 = \text{ReLU}(W_1 h_0 + b_1) \\ h_2 = W_2 h_1 + b_2 \\ h_3 = \text{Dropout}(h_2, p = 0.5) \end{gathered} \tag{16}$$

where h_i is the activation output of the i-th layer, and W_i and b_i are the weight matrix and bias vector of the i-th layer, respectively. After the output of the two fully connected layers, a Dropout layer is added with a dropout probability of 0.5, randomly dropping 50% of the neurons during training to prevent overfitting.

For the input layer h_0 of the MLP, the input data is x_{feature}:

$$h_0 = x_{\text{feature}} \tag{17}$$

For the output layer of the MLP, the final three-dimensional discrete feature vector z is obtained:

$$z = h_L \tag{18}$$

where L is the number of MLP layers. Through this process, the model effectively transforms continuous prior feature data F_k into discrete prior feature vectors z, enabling the use of these feature vectors for more complex analysis and processing in subsequent tasks.

C. Feature Concatenation

Feature concatenation combines different features, allowing the model to consider information from multiple feature levels simultaneously. This paper concatenates the discrete feature vector z generated by the MLP with the table sequence embedding E generated by the BERT input embedding layer to form the concatenated vector v. This concatenated vector v is then passed to the subsequent encoding layer. The shape of the discrete feature vector z is (k, dz), where dz is the embedding dimension of the MLP, which is the same as the embedding dimension dx of BERT, both being 768. This discrete feature vector z can be horizontally concatenated with the word vectors E (with shape $(n \times m, dx)$) generated by BERT Embeddings to produce the concatenated representation v, with shape $(m \times n + k, dx)$. The concatenation process can be represented as:

$$v = [E; z] \tag{19}$$

where $[\cdot;\cdot]$ denotes the vertical concatenation operation for vectors or matrices.

VI. Experimental Results and Analysis

A. Experimental Design

1) Experimental Environment

The hardware platform used for this deep learning experiment includes: an AMD 5600 series central processing unit (CPU), an NVIDIA GeForce 1660 Super graphics processing unit (GPU), and 32GB of system memory. The software environment is built on Conda, with the operating system being Ubuntu 20.04. The deep learning framework is PyTorch version 1.13.1 with CUDA 11.6 support, and the programming language environment is Python 3.9.18.

2) Datasets and Evaluation Metrics

Datasets: The experiment utilizes a private Internet of Vehicles (IoV) dataset and the public VizNet dataset. The IoV dataset consists of vehicle data collected for a specific automotive project, totaling 1,430 tables with 54 column types. The VizNet dataset is a subset of the original VizNet corpus, which is a collection of WebTables. This dataset is designed for the column type prediction task. It contains 78,733 tables and 119,360 columns annotated with 78 column types. The dataset is constructed by mapping column headers to DBpedia types using a set of mapping rules. This paper uses the same split for cross-validation to ensure that the evaluation results are directly comparable. Each column has only one label, making this task a multi-class classification task.

Evaluation Metrics: This paper uses micro-F1 and macro-F1 as evaluation metrics for model assessment and analysis.

(1) micro-F1: Micro-averaged F1 score. The micro-F1 score is a metric in statistics used to measure the accuracy of a binary classification model, particularly for imbalanced data. The micro F1 score comprehensively considers the true positives (TP), false negatives (FN), and false positives (FP) across all categories, calculating precision and recall through global aggregation. It simultaneously takes into account both the precision and recall of the classification model. Its formula is as follows:

$$\text{Recall}_m = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i + FN_i)} \tag{20}$$

$$\text{Precision}_m = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n}(TP_i + FP_i)} \tag{21}$$

$$\text{micro-F1} = 2 \times \frac{\text{Recall}_m \times \text{Precision}_m}{\text{Recall}_m + \text{Precision}_m} \tag{22}$$

where TP_i represents the number of true positives for the i-th class, and FN_i and FP_i represent the number of false negatives and false positives for the i-th class, respectively. If micro-F1 is closer to 1, it indicates that the model's recognition accuracy is higher. A higher micro-F1 value suggests that the model has good overall classification capabilities.

(2) macro-F1: Macro-averaged F1 score. The macro-F1 score independently calculates the F1 score for each class and then takes the average, ensuring that each class contributes equally regardless of the number of true samples. It assigns equal weight to each class, making it suitable for multi-class problems and unaffected by data imbalance. However, it can be influenced by classes with high recognition performance (high recall and precision). Its formula is as follows:

$$\text{F1-score}_i = 2 \times \frac{\text{Recall}_i \times \text{Precision}_i}{\text{Recall}_i + \text{Precision}_i} \tag{23}$$

$$\text{macro-F1} = \frac{\sum_{i=1}^{n} \text{F1-score}_i}{n} \tag{24}$$

where Recall_i and Precision_i represent the recall and precision for the i-th class, respectively. If macro-F1 is closer to 1, it indicates better recognition accuracy and comprehensiveness of the model, and vice versa.

Comparison Models: To fully validate the effectiveness of the proposed model, SATO, and DODUO are selected as comparison algorithms.

SATO: A hybrid machine learning model designed to automatically detect the semantic types of columns in tables by leveraging signals from the table context as well as the column values.

DODUO: A multi-task learning framework based on a pre-trained language model (called Doduo) that takes the entire table as input and uses a single model to predict column types and relationships.

B. Result Analysis

To validate the effectiveness of the proposed scheme, the experimental results are analyzed from the following aspects.

1) Experimental Results and Analysis

Table II presents a comparison of the experimental results of the proposed model (EFemb) with other existing models on the public dataset VizNet, with the best results highlighted in bold. Table III lists the top ten data types with the most significant improvement in F1 scores, most of which belong to minority classes. From Table III, it can be observed that the proposed model

(EFemb) outperforms most existing methods, particularly achieving a 1% improvement in the macro-F1 metric. Macro-F1 focuses on the average performance across all classes and is not affected by differences in the number of samples within each class. Notably, the VizNet dataset includes some minority class data such as "code," e.g., "5678 5675 TF-RS-4-9 06904BVK- FB TFG TRS 703-3 TFS-703HT." By incorporating prior features, the model can better understand and process information such as hyphens ('-') in such data. Consequently, the model's performance improves for these minority classes, which is also reflected in the increase in the macro-F1 score.

TABLE II. COMPARISON OF MACRO-F1 AND MICRO-F1 MODELS ON THE VIZNET DATASET

Model	Macro-F1 (%)	Micro-F1 (%)
Sato	65.35	89.32
Doduo	76.69	95.83
EFemb	77.61	95.30

TABLE III. THE TOP 10 CATEGORIES OF DATA WITH THE HIGHEST F1 SCORE IMPROVEMENT

Type	Support	F1	Improvement
person	10	1.00	1.00
manufacturer	10	1.00	0.3333
location	49	0.7692	0.2692
code	35	1.00	0.0909
weight	75	1.00	0.0667
symbol	30	0.6667	0.0667
category	158	0.9285	0.0320
rank	487	0.97	0.0096
description	265	0.71	0.0046
position	332	0.9394	0.0036

2) Ablation Experiment Analysis

Figure 3 and Table IV present the results of the model ablation experiments. To validate the effectiveness of the proposed method that integrates prior knowledge and prior feature embedding, the prior feature embedding module is used as the ablation variable in these experiments. Ablation experiments are conducted on both the Internet of Vehicles (IoV) dataset and the VizNet dataset. The prior feature vectors are directly input into BERT and concatenated with the original word vectors. After obtaining the final text representation in the embedding module, classification predictions are made.

As shown in Figure 3, the model incorporating prior feature embedding (FEemb) demonstrates significant advantages from the initial stages. It consistently outperforms the baseline model (origin) that does not include prior feature embedding. This advantage remains stable throughout the subsequent training process. By observing the trends of macro-F1 and micro-F1 scores with the number of iterations, it is clear that the black line (FEemb) in the figures always remains above the red line (origin). The performance gap between the two models exists from the first epoch and remains stable as training progresses, without significant narrowing. This phenomenon fully demonstrates that the prior feature embedding method not only has stronger representational capabilities during the model initialization phase but also consistently and stably improves model performance throughout the entire training process.

TABLE IV. COMPARISON OF MACRO-F1 AND MICRO-F1 ABLATION EXPERIMENTS USING DIFFERENT DATASETS

Dataset	Model Method	macro-F1(%)	micro-F1(%)
IoV	origin	86.31	91.27
	EFemb	94.23	96.35
VizNet	origin	76.69	95.83
	EFemb	77.61	95.30

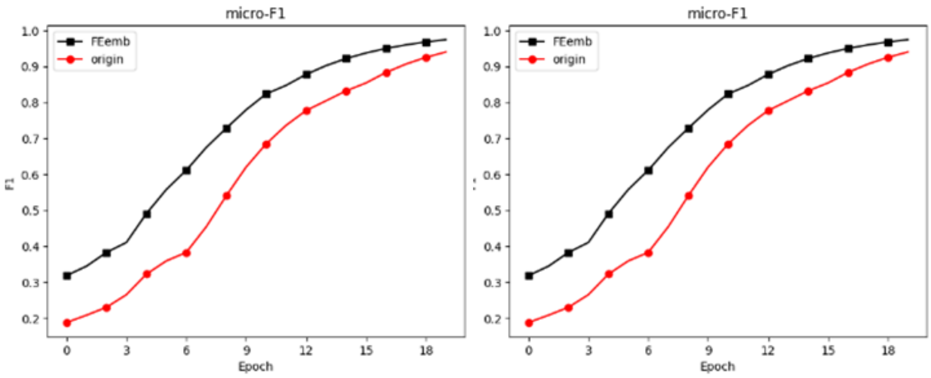

Fig. 3. Comparison of Ablation Experiments on the IoV Dataset.

TABLE V. USING DIFFERENT PRIOR FEATURE NUMBERS FOR MACRO-F1 AND MICRO-F1 ON VIZNET

Prior Feature Number	macro-F1(%)	micro-F1(%)
10 dims	77.61	95.30
20 dims	78.82	95.04
30 dims	79.30	95.04
50 dims	76.80	95.37

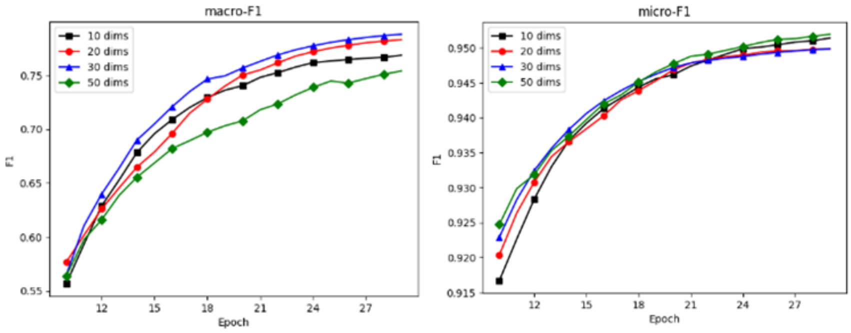

Fig. 4. Comparison of different prior feature numbers on VizNet.

3) Analysis of Experiments with Different Numbers of Prior Features

Table V and Figures 4 present the experimental results and process of the proposed model with different numbers of prior features. As shown in the table and figures, after incorporating prior feature embedding, the proposed model achieves a 1% ~ 3% improvement in macro-F1 for feature counts of 10, 20, and 30 compared to the model without prior feature embedding. As the number of features increases, the model's performance improves, but a performance decline is observed when the number of prior features reaches 50. This indicates that an appropriate number of prior features can significantly enhance model performance, but excessive features may lead to diminishing marginal returns.

VII. CONCLUSION

This paper proposes a vehicle data asset identification method based on pattern matching. The automated identification of vehicle data assets is achieved through pattern matching techniques. In the pattern matching process, a pretrained language model is utilized, incorporating prior feature embedding of prior knowledge to acquire external prior knowledge related to vehicle data. This enhances the semantic information of individual characters in the data and further applies the extracted statistical feature information to the classification of multi-data item sequences. Experimental results on the public dataset VizNet and a private dataset validate the effectiveness of the proposed method. The method effectively captures the correlation between prior features and vehicle data sequences, learns enhanced feature representations, and improves the model's predictive performance.

VIII. Author contributions

JiamingChen: Writing:original draft, Methodology, Investigation. Xiangyang Wang: Conceptualization, Supervision, Project administration. Chengyu Tan, Wei Luo & Changyun Huang:Resources, Data curation, Validation. All authors reviewed the manuscript.

IX. Funding

This work was supported by the State Key Laboratory of Intelligent Vehicle Safety Technology (Grant No. IVSTSKL-202314).

REFERENCR

[1] Y. Wang and H. Zhang, Data asset value assessment literature review and prospect, in *Proc. J. Phys.: Conf. Ser.*, vol. 1550, no. 3, 2020, Art. no. 032133.

[2] J. P. S. Medeiros, A. M. Brito, and P. S. M. Pires, A data mining based analysis of nmap operating system fingerprint database, in *Computational Intelligence in Security for Information Systems: CISIS'09, 2nd Int. Workshop*, Burgos, Spain, Sep. 2009, pp. 1--8.

[3] A. Rodriguez and K. Okamura, Cybersecurity text data classification and optimization for CTI systems, in *Web, Artificial Intelligence and Network Applications: Proc. Workshops 34th Int. Conf. Adv. Inf. Netw. Appl. (WAINA-2020)*, 2020, pp. 410--419.

[4] C. Li, P. Liu, and G. Cai, Research on dynamic network asset monitoring based on traffic awareness, *Inf. Secur. Res.*, vol. 6, no. 6, pp. 523--529, 2020.

[5] J. L. Bitzan, Determining critical information asset identification, classification, and valuation, Ph.D. dissertation, Capella Univ., 2020.

[6] D. D. Nucera, W. Quadrini, L. Fumagalli, *et al.*, Data-driven state detection for an asset working at heterogeneous regimens, *IFAC-PapersOnLine*, vol. 54, no. 1, pp. 1248--1253, 2021.

[7] S. Guo, Z. Guo, and Z. Qiu, HSMA: A pattern hierarchical matching algorithm for heterogeneous data in the Internet of Things, *J. Comput. Res. Dev.*, vol. 55, no. 11, pp. 2522--2531, 2018.

[8] Y. Li, D. Liu, and W. M. Zhang, A method for automatic schema matching using characteristic of data distribution, *Comput. Sci.*, vol. 32, no. 11, pp. 85--86, 2005.

[9] B. Yu, S. Tang, P. Zhang, *et al.*, A database schema matching method based on entity classification, *Comput. Sci.*, no. 10, pp. 157--159, 210, 2004.

[10] P. Yin, G. Neubig, W. Yih, *et al.*, TaBERT: Pretraining for joint understanding of textual and tabular data, *arXiv preprint arXiv:2005.08314*, 2020.

[11] Y. Suhara, J. Li, Y. Li, *et al.*, Annotating columns with pre-trained language models, in *Proc. 2022 Int. Conf. Manage. Data*, 2022, pp. 1493--1503.

[12] X. Deng, H. Sun, A. Lees, *et al.*, Turl: Table understanding through representation learning, *ACM SIGMOD Rec.*, vol. 51, no. 1, pp. 33--40, 2022.

[13] Y. Xiong, Y. Feng, H. Wu, *et al.*, Fusing label embedding into BERT: An efficient improvement for text classification, in *Findings Assoc. Comput. Linguistics: ACL-IJCNLP 2021*, 2021, pp. 1743--1750.

[14] M. Hulsebos, K. Hu, M. Bakker, *et al.*, Sherlock: A deep learning approach to semantic data type detection, in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 1500--1508.