Annotating Table Metadata with Knowledge-Enhanced Pre-trained Language Model

Qiyuan Zhang[1], Yuhua Tang[1(✉)], Jinlong Tian[1], Mengmeng Li[2], and Xudong Fang[2]

[1] National University of Defense Technology, Changsha, China
{zhangqiyuan,tianjinlong}@nudt.edu.cn,
yhtang62@163.com
[2] Academy of Military Sciences, Beijing, China
{lmm012328,fangxudong_nudt}@163.com

Abstract. The exponential growth of table data across various domains has underscored the critical need for accurate annotation of table metadata, particularly column types and inter-column relationships. Addressing this challenge, we introduce KETAM, a Knowledge-Enhanced Pre-trained Language Model that integrates domain-specific knowledge with pre-trained language models to enhance the precision and effectiveness of table metadata annotation. KETAM employs a novel methodology that fortifies table data with domain insights, leveraging these to annotate column types and inter-column relationships with unprecedented accuracy. Through extensive testing on the WikiTable and VizNet datasets, KETAM demonstrates superior performance, achieving a Macro F1 score of 86.62% and a Micro F1 score of 94.37% for column type annotation on the WikiTable dataset, and a Macro F1 score of 84.68% and a Micro F1 score of 93.26% on the VizNet dataset, significantly outperforming existing models like TURL and Sato. Similarly, for column relationship annotation on the WikiTable dataset, KETAM obtains a Macro F1 score of 84.91% and a Micro F1 score of 93.70%, surpassing TURL by a notable margin. These robust empirical results validate KETAM's superior performance and reliability in annotating table metadata, showcasing its potential to revolutionize data analysis through advanced table understanding.

Keywords: Knowledge Enhancement · Pre-training · Tabular Data.

1 Introduction

The metadata of tables, such as column types and inter-column relationships, has served as the backbone for effectively understanding tabular data, and played a critical role in various data management tasks [13,14,39]. Specifically, column types provide a semantic understanding of each column, which support data validation [30], query optimization [12], and automated report generation [9],

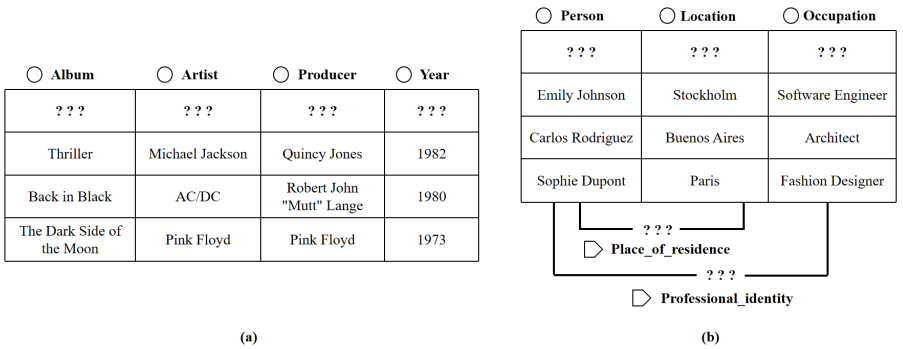

Fig. 1. Two table metadata annotation tasks. (a) Annotate the column type based on the cell value in the table, the column types are marked above the table. (b) Annotate the relationship between two columns, Inter-column relationships are displayed below the table.

etc. Similarly, inter-column relationships are indispensable for data management tasks, including data integration [2], schema mapping [28], and knowledge graph construction [16], which reveal intrinsic linkages among disparate data elements and enable sophisticated analysis across various tabular data sources.

Despite the importance of semantic column types and column relationships of tables, these metadata elements are often missing or incomplete due to factors such as data collection methods that prioritize efficiency over detail, the dynamic nature of data sources, and the sheer volume of data that makes manual curation impractical [20,10]. Manual identification and supplementation of these metadata are not only costly and time-consuming but also require a substantial amount of domain expertise, making it an inefficient process for large-scale data management [27]. Consequently, there remains urgent need for automatically annotation and enrichment for table metadata.

Example 1. Figure 1 presents two tables with missing column type and relationship annotations. The table in Fig. 1(a) enumerates various music albums along with their artists, producers, and release years. Within the second and third columns, person names require contextual cues within the same column to ascertain the correct column types. For instance, "Pink Floyd" appears in both columns, necessitating an examination of other names in the columns to grasp their semantic implications. Furthermore, column types are sometimes contingent upon other columns within the table. By taking into account contextual information, models can deduce that the table's theme revolves around music albums, thereby understanding that the second and third columns are unlikely to be directors or athletes, and the fourth column is most probably indicative of the album's release year. Figure 1(b) depicts a table concerning individuals, listing persons, locations, and occupations. The column types Person and Location are instrumental in predicting the "Place_of_residence" relationship; however,

additional contextual information is imperative to discern whether the locations correspond to "Place_of_residence" or "Place_of_birth". Distinguishing between these two types of locations requires a deeper understanding of the table's context.

These examples underscore the importance of understanding intra-column, inter-column, and whole-table contexts for precise prediction of column types and relationships, a challenge for traditional machine learning models. While models like BERT [8] and GPT [26] excel in capturing context from unstructured text, their application to structured data is underexploited. Current approaches to integrating table data into pre-training are not fully effective, and more advanced models are needed to harness table information. Additionally, annotating table data is complex due to cell multi-entity correspondence, requiring precise model disambiguation. Accurate interpretation of table contexts is essential for correct type and relationship assignment, hindered by the variability and inconsistencies in real-world tabular data.

In this paper we propose a framework that effectively annotate tabular data (KETAM). Specifically, we first serialize table data into text sequences and augment them with triples from an external knowledge base to form a knowledge-enhanced table tree. Secondly, we utilize a pre-trained language model to transform embeddings and mask irrelevant knowledge using a masked matrix. Thirdly, we employ a Transformer encoder to learn hidden states and predict column types and inter-column relationships. We extract standard types, relationships, and entities from the YAGO [31] catalog, which encompasses approximately 250,000 types, 2 million entities, and 99 relationships. Utilizing ground truth synthesized from Wikipedia [35], DBPedia [1], and YAGO, we train our model and evaluate the accuracy of the annotation tasks. Experimental results demonstrate that KETAM achieves outstanding performance on the WikiTable and VizNet datasets, significantly outperforming existing models in both column type annotation and column relationship annotation tasks.

To sum up, we make the following contributions in this paper:

- We emphasize the importance of integrating domain-specific knowledge into pre-trained language models to enhance the accuracy and effectiveness of annotating table metadata, including column types and inter-column relationships.
- We introduce KETAM, a novel framework that fuses knowledge-enhanced techniques with pre-trained language models, thereby elevating the annotation of table metadata to new heights of accuracy.
- We conduct comprehensive experiments on two real-world datasets, validating the effectiveness of our KETAM model. Through extensive testing and evaluation, we provide strong empirical evidence supporting the superior performance and reliability of KETAM.

2 Related Work

Method based on knowledge base. Existing column type prediction models address column type prediction by modeling it as a multi-classification task using machine learning techniques. Hulsebos et al. [14] developed a deep learning model called Sherlock, which detects the semantic types of data columns by training a multi-input neural network on a large corpus of real-world data, utilizing features like character distributions, word embeddings, and paragraph vectors to predict column types with high accuracy. Zhang et al. [39] proposed Sato, a hybrid machine learning model that detects the semantic types of table columns by leveraging both the contextual signals from the table and the column values, achieving state-of-the-art performance in semantic type prediction. Recent advancements have integrated knowledge bases to enhance performance, leveraging entity matching, hierarchical relationships, and graphical models to provide context and disambiguation cues [32,17]. Deep learning models like Hybrid Neural Networks (HNN) [6] and ColNet [5] have further improved accuracy by combining cell embedding, table locality feature learning, and property features extracted from KBs. These models use external knowledge bases (KB) to improve machine learning models and have achieved success in column type prediction tasks, outperforming classical machine learning models.

Another task is to annotate the column relations between pairs of columns in the same table, that is, the semantic labels between a pair of columns in a table [23,34,4]. Limaye et al. [19] proposed machine learning techniques to annotate table cells with entities, columns with types, and relationships between column pairs. Their work introduced a graphical model that simultaneously infers entities, types, and relations. The advent of large-scale knowledge graphs (KGs) has provided a rich resource for semantic annotation of web tables. Ritze et al. [29] presented a method to match web tables to DBpedia, using the knowledge graph's ontology to infer column types and relationships. Building on this, Cannaviccio et al. [3] introduced an approach that does not require entity linking to a KG. Instead, they utilized generative language models derived from web-scale corpora to rank relations between entities mentioned in table cells. Macdonald et al. [21] explored the use of Long Short-Term Memory (LSTM) networks for predicting relationships between column pairs in Wikipedia tables. By training on annotated tables, the LSTM model learned to associate table structures with specific relations, achieving high accuracy in predicting relations even in the absence of a KG.

Method based on pre-trained language model. The recent surge in pre-trained language models has profoundly impacted natural language processing, particularly in semantic understanding and information extraction [18,15,25]. This progress has extended to table understanding with notable advancements. For instance, TURL [7] leverages a structure-aware Transformer encoder and a masked entity recovery pre-training objective to learn deep contextualized representations for relational Web tables. Trabelsi et al. [33] introduced SeLaB, a

context-aware semantic labeling method utilizing BERT to generate schema labels for data tables by integrating column values and contextual information from other columns. TaBERT [38] jointly learns representations for natural language sentences and structured tables, enabling improved performance on semantic parsing tasks involving both textual and tabular data. TUTA [36] employs tree-based transformers to capture spatial, hierarchical, and semantic information in variously structured tables, achieving state-of-the-art performance on tasks like cell type classification and table type classification.

Existing table understanding models primarily employ either knowledge-based or pre-trained language model-based approaches. Despite the significant success of Large Language Models (LLMs) in processing natural language [24], their capacity to understand and generate annotations for tabular data remains limited. Knowledge-based methods excel in precision but suffer from low recall due to the limitations of knowledge bases. Conversely, language model-based approaches offer higher recall at the cost of some precision. The optimal balance between these two approaches remains an open research question [11]. Our proposed KETAM model addresses this challenge by effectively fusing domain knowledge to enhance the input of pre-trained language models, achieving improved performance.

3 Method

In this section, we detail the implementation of KETAM, as shown in Fig. 2.

3.1 Knowledge Injection

The knowledge injection module within the KETAM framework transforms tabular data into token sequences. Recognizing the reliance of state-of-the-art models on sequential text, KETAM employs a straightforward yet remarkably efficient approach: it concatenates the cell values from each column into a sequential chain, prepending this sequence with a distinctive $[CLS]$ token. This prefix serves as a marker to denote the start of the contextual sequence.

In certain scenarios, the method of converting cell values into text may be constrained due to the nature of the data within table columns. Columns may be of numeric types, such as floating-point numbers or dates, which inherently lack semantic information. Directly converting such values into embedding vectors may not capture the complex relationships and patterns within the data [22]. However, recent advancements have seen the expansion of Transformer models to accommodate numerical data [37]. In Section 4.2, we provide a brief analysis of KETAM's performance on numerical column types.

Consider a table T, comprising n columns, where KETAM's serialization mechanism systematically generates n distinct token sequences. Let us delve into one such sequence, denoted as $s = [v_0, v_1, v_2 \ldots, v_m]$, where m signifies the count of rows in the table. To imbue this sequence with domain-specific knowledge, KETAM harnesses an external knowledge base $\mathbb{K}$. For each cell value

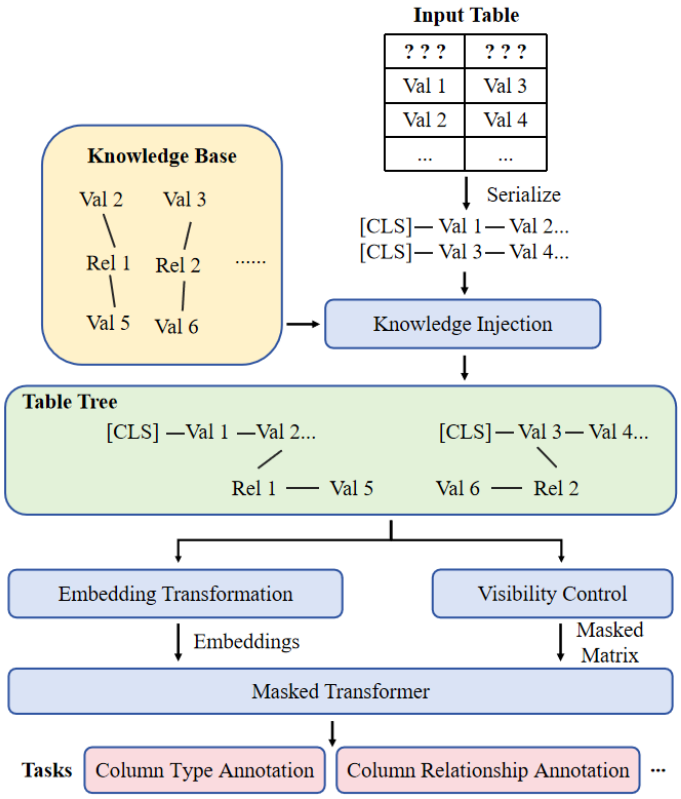

Fig. 2. Overview of KETAM's model architecture. KETAM augments the input to pre-trained language models with external domain-specific knowledge bases. We can select the appropriate knowledge base for table data sets from different domains, thereby enhancing the semantic understanding and annotation accuracy of the table contents.

$[v_i]$ in the sequence, a targeted extraction process retrieves a collection of relevant triples $E = [(v_i, r_{i0}, v_{i0}), \ldots, (v_i, r_{ik}, v_{ik})]$ and each triple encapsulates pertinent information associated with the entity represented by $[v_i]$. Subsequently, these extracted triples are injected into the original sequence s, yielding a knowledge-augmented table tree representation t. The structure of t is shown in Fig. 3.

3.2 Embedding Transformation

The embedding transformation component of KETAM is responsible for converting the knowledge-enhanced table tree into a vector representation that can be processed by the model. Similar to BERT, KETAM employs a three-component embedding scheme: token embedding, position embedding, and segment embedding.

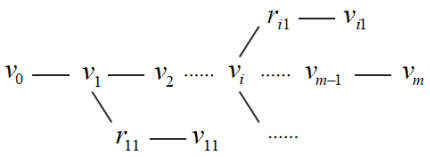

Fig. 3. Structure of the table tree.

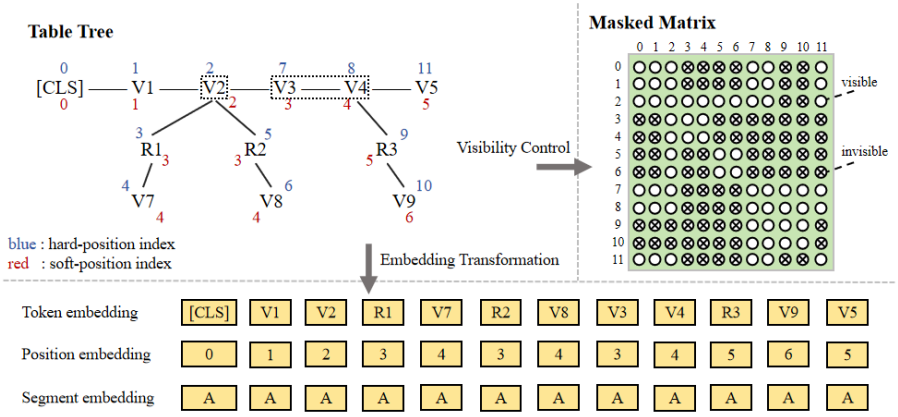

Fig. 4. Specific process of embedding transformation and visibility control. In the table tree, the blue numbers correspond to the hard position index of each token, and the red numbers correspond to the soft position index of each token. The tokens in the table tree are flattened into a sequence of token embeddings through the hard position index, and the soft position index is used as the position embedding. All tokens in the first column are labeled as "A". In the masked matrix, the hollow circles represent visible and the circles with $\times$ represent invisible.

Token embedding. KETAM convert each token in the table tree into an H-dimensional embedding vector. Tokens representing injected knowledge triples, i.e., the branch structure in the table tree, are placed after the corresponding entity tokens, and the order of the following tokens in the original sequence is adjusted backwards one by one. For example, in Fig. 4, the table tree is re-ordered as $[[CLS], V_1, V_2, R_1, V_7, V_3, V_4, V_5, R_2, V_8, R_3, V_9, V_6]$. This reordering is necessary to preserve the semantics of the table tree, but it disrupts the original sequence order, potentially leading to misinterpretations of the underlying semantics. These potential semantic errors will be addressed in the subsequent visibility control component.

Position embedding. KETAM encodes the position information of each token within the table tree, adding positional context to the sequence. There are two primary indexing methods for position embedding: hard and soft indexing. Hard

indexing assigns a fixed position number to each token based on its order in the sequence, representing absolute positions. Soft indexing, on the other hand, dynamically adjusts position information based on sentence structure and injected knowledge, preserving hierarchical and nested structure information and representing relative positions. KETAM employs soft indexing due to the presence of knowledge-enhanced table trees. This allows the model to retain structural information even after reordering and helps distinguish directly related tokens from indirectly related ones through knowledge injection. However, challenges remain. For instance, in Fig. 4, $[R_1]$ and $[V_3]$ have the same position number 3, despite being unrelated. This will be solved in the visibility control section later.

Segment embedding. The segment embedding component is crucial when dealing with multiple columns, as it helps the model identify which column each token belongs to. This is achieved by adding a segment marker to each token, effectively distinguishing tokens from different columns. For example, when processing two columns, all tokens from the first column are marked as segment $[A]$, while those from the second column are marked as segment $[B]$.

3.3 Visibility Control

Injecting external knowledge into a model can potentially distort the semantic integrity of the inherent sequence representation. Specifically, as exemplified in Fig. 4, where relations such as $[R_1]$ and entities like $[V_7]$ are intrinsically tied to $[V_2]$ but not necessarily to other entities, it is imperative to ensure that the embeddings of these unrelated entities remain unaffected by the injected information. To address this challenge, we devise a novel visibility control mechanism by introducing a masked matrix M that meticulously regulates the inter-token visibility, thereby refining the computation of self-attention.

The masked matrix M is defined as (3),

$$M_{ij} = \begin{cases} 0 & v_i \ominus v_j \\ -\infty & v_i \oslash v_j \end{cases} \tag{1}$$

M_{ij} is set to 0 if tokens i and j belong to the same semantic or structural branch, fostering their mutual visibility and interaction. Conversely, M_{ij} is assigned a negative infinity value, effectively severing the connection and ensuring invisibility between tokens that are not directly relevant, as depicted by the white dots in Fig. 4. This approach ensures that the injected external knowledge, though pertinent to its head entity, does not inadvertently influence unrelated tokens, thus preserving the model's focus solely on pertinent information.

3.4 Attention Masking

For each element in a sequence, the model generates three vectors: Query (Q), Key (K), and Value (V). These vectors are obtained by multiplying the element's embedding vector with different weight matrices. The self-attention mechanism

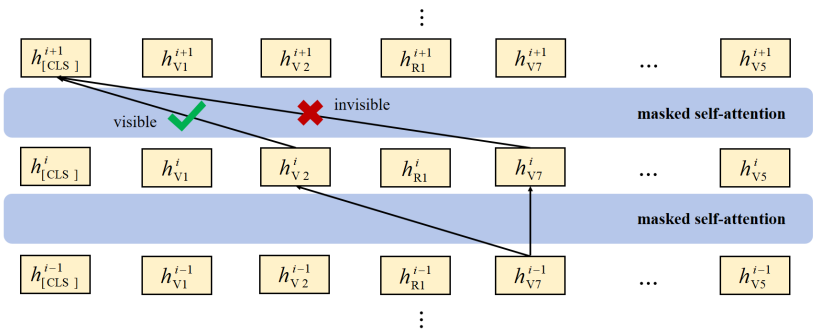

Fig. 5. Illustration of the Masked Transformer, consists of multiple masked self-attention blocks.

computes a score matrix by taking the dot product of the Query vector of each element with the Key vectors of all elements in the sequence. This score matrix represents the level of attention each element pays to every other element. The normalized attention scores are then used as weights to perform a weighted sum of the corresponding Value vectors, resulting in a final output vector that is a transformed representation of the input element, incorporating information from all other elements in the sequence.

However, since the Transformer encoder in BERT cannot accept a masked matrix as input, we modify it to a masked Transformer that restricts the self-attention scope according to the masked matrix M. Specifically, when calculating self-attention, if an element M_{ij} in the masked matrix is 0, the corresponding attention score is set to a very small value. By adding the score matrix to the masked matrix, after applying the Softmax function, the weight of that term will be close to zero. The masked Transformer is composed of multiple masked self-attention modules stacked together. As shown in Fig. 5, which depicts a part of the masked Transformer structure, h^i represents the hidden state of the i-th masked self-attention block. It can be observed that $h^i_{[V_7]}$ does not affect $h^{i+1}_{[CLS]}$ because $[V_7]$ is not visible to $[CLS]$. However, $h^{i+1}_{[CLS]}$ can indirectly obtain information from $h^{i-1}_{[V_7]}$ through $h^i_{[V_2]}$, because $[V_7]$ is visible to $[V_2]$, and $[V_2]$ is visible to $[CLS]$. The benefit of this process is that $[V_7]$ enriches the embedding representation of $[CLS]$ without altering the meaning of the original sequence.

4 Experiment

4.1 Experimental Settings

Dataset. We utilized two standard datasets for our evaluation:

– **WikiTable** [7]: A large dataset extracted from Wikipedia, comprises approximately 1.65 million tables spanning diverse topics. For column type annotation, we selected columns with at least three Freebase entity links,

resulting in 255 types across 628,254 columns in 397,098 tables. For column relationship annotation, we focused on pairs with a majority of entity pairs sharing a common relationship, identifying 121 types across 62,954 pairs in 52,943 tables.

– **VizNet** [39]: A large-scale visualization training dataset, from which we utilized a subset for our study, specifically the WebTables dataset within the VizNet corpus [13]. We selected relational Web tables with valid headers from the WebTables corpus that fall within 78 predefined semantic types, which are derived from the T2Dv2 Gold Standard. To avoid filtering out columns due to minor variations in casing or representation, we normalized the column headers. The VizNet dataset was used exclusively for the column type annotation task, where we annotated 119,360 columns across 78,733 tables.

Baselines. We compare KETAM with the following baseline models:

– **TURL** [7]: The first framework that introduces pre-training and fine-tuning for relational Web tables, featuring a structure-aware Transformer to encode row-column structures and a Masked Entity Recovery (MER) objective for pre-training on unlabeled data. This approach enables broad generalization across diverse table tasks with minimal fine-tuning.
– **Sherlock** [14]: A deep learning model for detecting semantic data types in tables. It employs a multi-input neural network trained on extensive real-world data, extracting over 1,500 features to integrate global statistics and contextual information, thereby accurately predicting semantic types and bolstering data analysis.
– **Sato** [39]: A hybrid machine learning model that detects semantic types of data columns in tables by leveraging both column values and table context. It employs deep learning, topic modeling, and structured prediction to resolve ambiguities and enhance type detection accuracy. Sato achieves state-of-the-art performance on the column type prediction task on the VizNet dataset.
– **ColNet** [5]: A neural network framework for column type annotation that integrates knowledge base inference with machine learning, automatically training Convolutional Neural Networks to predict column types. It captures intra-cell contextual semantics and inter-cell column semantics, achieving precise type prediction without reliance on table metadata.

Evaluation Metrics. We employed Macro F1 and Micro F1 as evaluation metrics. Macro F1 was calculated by determining the F1 score for each category individually and then averaging these scores, without accounting for the significance or sample size variations among categories. In contrast, Micro F1 was based on a weighted average of the F1 values, taking into consideration the sample size of each category, thereby assigning greater weight to categories with a larger number of samples.

Implementation Details. The number of masked self-attention layers and heads was set to 12, and the hidden dimension of the embedding vectors was configured to 768. During the pre-training phase, all settings for KETAM were consistent with BERT's. The model was trained for 30 epochs, and the checkpoint with the highest F1 score on the validation set was selected for further evaluation.

For column type annotation, KETAM incorporates a dense layer sized to the number of column types, enabling it to output a probability distribution for potential types and facilitating type-specific predictions. In column relationship annotation, KETAM concatenates contextual representations from pairs of columns and feeds this into an additional dense layer to capture their combined information.

4.2 Results

Column Type Annotation. Table 1 delineates the experimental outcomes for the column type annotation task, leveraging the WikiTable and VizNet datasets. Our proposed KETAM model demonstrates a marked superiority over existing state-of-the-art methodologies across both datasets. Specifically, on the WikiTable dataset, KETAM procures a notable enhancement of 6.2% in Micro F1 and 10.2% in Macro F1 relative to TURL. This performance increment is mirrored on the VizNet dataset, where KETAM outperforms Sato by 5.4% in Micro F1 and 11.9% in Macro F1.

It warrants emphasis that the structural divergence of the VizNet dataset from that of the TURL model's pre-training corpus precludes a direct comparative analysis. Conversely, the inapplicability of the WikiTable dataset to Sato is attributed to its distinct structural requirements. The TURL model, which is contingent upon supplementary metadata such as table titles and column headers for peak performance, was subjected to testing within a metadata-rich environment. Nonetheless, our findings reveal that TURL's performance does not eclipse that of KETAM, even when augmented with metadata, thus underscoring the merit of integrating external knowledge within our framework.

The observed tendency for Macro F1 scores to trail Micro F1 scores is primarily ascribed to the diminished F1 scores associated with less prevalent column types. This variance accentuates the intricacies inherent in the precise annotation of infrequent column types and accentuates the necessity for a balanced dataset composition to ensure a holistic assessment of model performance.

Additionally, we conducted a comparative study on the application of large language models (LLMs) for table annotation tasks. We devised a simple prompt, informing the LLMs of the semantic types of all columns initially, followed by directly inputting the column data to query the model's prediction of the column's semantic type. Due to the extensive size of the original dataset, we randomly selected 10,000 columns for testing, utilizing GLM-4 [40] and GPT-4o as our LLMs of interest. The results, as presented in Table 2, demonstrate that both GLM-4 and GPT-4o significantly underperformed compared to KETAM on the

Table 1. Performance on the Column Type Annotation

Method	WikiTable		VizNet	
	Macro F1	Micro F1	Macro F1	Micro F1
Sherlock	70.24	78.83	69.28	86.71
ColNet	71.61	82.91	73.19	88.26
TURL	78.61	88.87	-	-
Sato	-	-	75.63	88.49
KETAM	**86.62**	**94.37**	**84.68**	**93.26**
TURL & metadata	80.61	92.69	-	-
KETAM & metadata	87.16	94.84	84.72	93.87

column type annotation task, thereby validating the effectiveness of our model in comparison to these large models.

Table 2. Comparison with Large Language Model

Method	WikiTable		VizNet	
	Macro F1	Micro F1	Macro F1	Micro F1
GLM-4	47.84	52.64	45.94	51.47
GPT-4o	56.37	64.38	52.78	60.34
KETAM	**87.32**	**94.64**	**85.14**	**94.18**

Column Relationship Annotation. Table 3 elucidates the empirical results for column relationship annotation on the WikiTable dataset. The VizNet dataset, which is constrained to providing annotations for column types, was deemed unsuitable for this specific task, hence, it was not leveraged in our analysis. Our KETAM model manifests a pronounced superiority over the prevailing benchmark, TURL, particularly in the domain of column relationship annotation. Specifically, KETAM procures a substantial enhancement of 3.5% in Micro F1 and 9.9% in Macro F1 compared with TURL.

Table 3. Performance on the Column Relationship Annotation

Method	P	R	Macro F1	Micro F1
TURL	91.13	90.58	77.29	90.57
KETAM	**93.94**	**93.47**	**84.91**	**93.70**
TURL & metadata	92.49	92.62	80.16	92.55
KETAM & metadata	94.13	94.72	85.79	94.42

While the Sherlock model, renowned for its efficacy in column type prediction, was not deemed pertinent for the column relationship prediction task.

Meanwhile, ColNet is specifically tailored for column type prediction and does not directly involve column relationship prediction. We undertook a meticulous fine-tuning of the TURL model to adeptly address the column relationship prediction. Supplemented with metadata to optimize its performance, TURL was rigorously tested. Nonetheless, the empirical evidence unequivocally demonstrates that KETAM outperforms TURL, thereby substantiating the efficacy of our proposed approach. This comparative analysis underscores the robustness of KETAM in discerning intricate column relationships, even in the absence of metadata, and reaffirms the model's potential in advancing the state-of-the-art in table annotation tasks.

Performance on Numerical Column Types. As mentioned in Section 3.1, KETAM's conversion of cell values into text sequences poses challenges for numerical column types. Table 4 presents KETAM's predictive accuracy on 12 numerical types from VizNet, including "plays", "fileSize", and "grades", which are represented as integers, floats, or dates. The table lists the numerical percentage (%num) for each type, which is the percentage of cell values in a column that can be identified as numerical. It is observed that KETAM underperformed on certain numerical types, such as "rankings". This may be due to these columns having less distinct numerical patterns or not matching the typical semantic features of numerical types.

Despite these challenges, KETAM achieves an average F1 score of 87.7% for numerical types, close to its overall Macro F1 score on VizNet (85.8%). This suggests that KETAM can recognize numerical patterns and predict column types effectively, likely due to the Transformer model's self-attention mechanism adept at capturing numerical data's structure and patterns. The robust performance on numerical columns indicates that KETAM's text-based approach does not impede its ability to identify and classify numerical types.

Table 4. KETAM's Performance on Numerical Column Types. %num represents how many cell values in a type are numeric types.

type	%num	F1	type	%num	F1
plays	100.00	89.94	elevation	87.39	92.37
rank	93.01	95.17	ranking	86.88	35.61
depth	92.86	88.19	age	81.04	97.40
sales	92.05	76.91	birthDate	67.85	96.17
year	91.47	98.13	grades	67.18	96.72
fileSize	87.84	89.19	weight	60.41	96.64

4.3 Ablation Studies

In this subsection, we delve into the analysis of the influence exerted by the knowledge base, soft position indexing, and the masked matrix on the perfor-

mance of the KETAM model. Given that KETAM is a knowledge-enhanced variant of BERT, we conduct a comparative study by employing BERT in its standard form, devoid of a knowledge base, to quantify the benefits of knowledge integration.

To ascertain the impact of soft position indexing, we conduct an ablation study where we replace it with hard position indexing during the embedding transformation phase. Additionally, to assess the significance of the masked matrix, we introduce a control by setting all tokens to be fully visible, thereby permitting unrestricted attention across tokens, which means that tokens can attend to irrelevant knowledge as well.

Table 5. Ablation study on the Column Type Annotation. The numbers in the figure represent the micro F1 scores of the annotation task.

Method	WikiTable	VizNet
KETAM	**94.40**	**93.49**
KETAM w/o soft-position	90.73	89.28
KETAM w/o masked matrix	88.94	87.64
BERT	82.17	80.34

Table 6. Ablation study on the Column Relationship Annotation.

Method	Macro F1	Micro F1
KETAM	**85.31**	**93.87**
KETAM w/o soft-position	78.72	89.43
KETAM w/o masked matrix	78.36	88.57
BERT	67.92	81.34

Tables 5 and 6 delineate the experimental outcomes of KETAM and its variants across the tasks of column type annotation and column relationship annotation, respectively. The base KETAM model secures the highest scores on both the WikiTable and VizNet datasets, thereby validating its efficacy in leveraging its comprehensive architecture for precise annotation tasks.

The results demonstrate a performance decrement when soft position indexing is omitted (KETAM w/o soft-position), highlighting the importance of dynamically adjusting positional information with integrated knowledge for maintaining precision. Additionally, excluding the masking matrix (KETAM w/o masked matrix) results in performance drops, underscoring its role in regulating token visibility during self-attention to prevent semantic misinterpretations. Overzealous knowledge injection without proper masking can lead to semantic deviations. Furthermore, a model without an external knowledge base (BERT) shows significantly lower scores, emphasizing KETAM's substantial improvement

over conventional pre-trained language models. This comparative analysis underscores the efficacy of employing a knowledge-enhanced pre-trained language model for the nuanced tasks of table annotation.

5 Conclusion

In this paper, we propose KETAM, an innovative framework designed for the automatic annotation of table metadata. It employs a knowledge-enhanced pre-trained language model to achieve superior accuracy and efficiency in the annotation of column types and inter-column relationships. By incorporating domain-specific knowledge into the model, KETAM not only enhances the understanding of tabular data but also paves the way for more nuanced data analysis methodologies.

Future research directions for KETAM include the investigation of diverse knowledge bases to enrich the model's capabilities, the expansion of annotation tasks to cover a broader spectrum of table structures, and the optimization of the framework for enhanced efficiency and scalability. Advancing in these areas will empower KETAM to tackle an expanded array of challenges in table understanding, thereby bolstering the evolution of advanced data analysis techniques.

References

1. Auer, S.e.a.: Dbpedia: A nucleus for a web of open data. In: ISWC '07. pp. 722–735. Springer (2007)
2. Cafarella, M.J.e.a.: Data integration for the relational web. Proc. VLDB Endow. **2**(1), 1090–1101 (2009)
3. Cannaviccio, M.e.a.: Towards annotating relational data on the web with language models. In: WWW '18. pp. 1307–1316 (2018)
4. Cappuzzo, R.e.a.: Creating embeddings of heterogeneous relational datasets for data integration tasks. In: SIGMOD '20. pp. 1335–1349 (2020)
5. Chen, J.e.a.: Colnet: Embedding the semantics of web tables for column type prediction. In: AAAI. vol. 33, pp. 29–36 (2019)
6. Chen, J.e.a.: Learning semantic annotations for tabular data. arXiv preprint (2019)
7. Deng, X.e.a.: Turl: Table understanding through representation learning. ACM SIGMOD Rec. **51**(1), 33–40 (2022)
8. Devlin, J.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint (2018)
9. Do, Q.e.a.: Automatic generation of sql queries. In: ASEE '14. pp. 24–221 (2014)
10. Doan, A.e.a.: Learning to map between ontologies on the semantic web. In: WWW '02. pp. 662–673 (2002)
11. Fan, G.e.a.: Table discovery in data lakes: State-of-the-art and future directions. In: SIGMOD '23 Companion. pp. 69–75 (2023)
12. Herodotou, H.e.a.: Query optimization techniques for partitioned tables. In: SIGMOD '11. pp. 49–60 (2011)
13. Hu, K.e.a.: Viznet: Towards a large-scale visualization learning and benchmarking repository. In: CHI '19. pp. 1–12 (2019)

14. Hulsebos, M.e.a.: Sherlock: A deep learning approach to semantic data type detection. In: KDD '19. pp. 1500–1508 (2019)
15. Jiang, Z.e.a.: How can we know what language models know? Trans. Assoc. Comput. Linguistics **8**, 423–438 (2020)
16. Kejriwal, M.: Domain-specific knowledge graph construction. Springer (2019)
17. Khurana, U.e.a.: Semantic concept annotation for tabular data. In: CIKM '21. pp. 844–853 (2021)
18. Li, Y.e.a.: Deep entity matching with pre-trained language models. arXiv preprint (2020)
19. Limaye, G.e.a.: Annotating and searching web tables using entities, types and relationships. Proc. VLDB Endow. **3**(1-2), 1338–1347 (2010)
20. Liu, Y.e.a.: Tableseer: automatic table metadata extraction and searching in digital libraries. In: JCDL '07. pp. 91–100 (2007)
21. Macdonald, E.e.a.: Neural relation extraction on wikipedia tables for augmenting knowledge graphs. In: CIKM '20. pp. 2133–2136 (2020)
22. Mikolov, T.: Efficient estimation of word representations in vector space. arXiv preprint **3781** (2013)
23. Muñoz, E.e.a.: Using linked data to mine rdf from wikipedia's tables. In: WSDM '14. pp. 533–542 (2014)
24. Naveed, H.e.a.: A comprehensive overview of large language models. arXiv preprint (2023)
25. Petroni, F.e.a.: Language models as knowledge bases? arXiv preprint (2019)
26. Radford, A.: Improving language understanding by generative pre-training (2018)
27. Rahm, E.e.a.: Data cleaning: Problems and current approaches. IEEE Data Eng. Bull. **23**(4), 3–13 (2000)
28. Rahm, E.e.a.: A survey of approaches to automatic schema matching. VLDB J. **10**, 334–350 (2001)
29. Ritze, D.e.a.: Matching html tables to dbpedia. In: WIMS '15. pp. 1–6 (2015)
30. Song, J.e.a.: Auto-validate: Unsupervised data validation using data-domain patterns inferred from data lakes. In: SIGMOD '21. pp. 1678–1691 (2021)
31. Suchanek, F.M.e.a.: Yago: a core of semantic knowledge. In: WWW '07. pp. 697–706 (2007)
32. Takeoka, K.e.a.: Meimei: An efficient probabilistic approach for semantically annotating tables. In: AAAI '19. vol. 33, pp. 281–288 (2019)
33. Trabelsi, M.e.a.: Selab: Semantic labeling with bert. In: IJCNN. pp. 1–8. IEEE (2021)
34. Venetis, P.e.a.: Recovering semantics of tables on the web (2011)
35. Völkel, M.e.a.: Semantic wikipedia. In: WWW '06. pp. 585–594 (2006)
36. Wang, Z.e.a.: Tuta: Tree-based transformers for generally structured table pretraining. In: KDD '21. pp. 1780–1790 (2021)
37. Wu, N.e.a.: Deep transformer models for time series forecasting: The influenza prevalence case. arXiv preprint (2020)
38. Yin, P.e.a.: Tabert: Pretraining for joint understanding of textual and tabular data. arXiv preprint (2020)
39. Zhang, D.e.a.: Sato: Contextual semantic type detection in tables. arXiv preprint (2019)
40. ZHIPU: ZHIPU AI DevDay GLM-4. URL: `https://zhipuai.cn/en/devday` (2024)