A Domain-Independent Approach for Semantic Table Interpretation

Binh Vu(✉), Craig A. Knoblock, and Fandel Lin

Information Sciences Institute and Department of Computer Science, University of Southern California, Marina del Rey, CA 90292, USA
{binhvu,knoblock,fandelli}@isi.edu

Abstract. Understanding the semantic structure of tabular data is essential for data integration and discovery. Specifically, the goal is to annotate columns in a tabular source with types and relationships between them using classes and predicates of a target ontology. Previous work either requires trained labeled data or exploits the overlapping data between the table data and a knowledge graph to predict types and relationships. However, these approaches cannot be used in a new domain with limited labeled data. To address this issue, we propose a novel domain-independent approach to estimate a score reflecting the semantic relatedness between a table column and an ontology class or property using the table metadata and data. Our empirical evaluation demonstrates that our approach significantly outperforms strong baselines based on large language models.

Keywords: Semantic Models · Semantic Descriptions · Data Integration · Knowledge Graphs · Semantic Web

1 Introduction

The semantic modeling problem, also known as semantic table interpretation, involves annotating the types and relationships of columns using classes and predicates from an ontology. The resulting annotation, called a semantic description, enables tabular data to be mapped to a common ontology, a key step in many data integration pipelines. This problem has become increasingly important in today's world, where machine learning applications, analytics, and predictive workflows rely on large volumes of high-quality, well-structured data.

Existing approaches to semantic modeling generally fall into two categories. The first group leverages overlapping data between the input tables and a target knowledge graph (KG) [8,17,19,23,24,29,30,33,45]. These methods typically identify column types and relationships by matching entities and their associated properties across columns. However, it is well known that KGs are incomplete and there are domains where KGs have very little data. The second category uses supervised learning techniques that rely on labeled data sources for training [11,37,40]. These methods do not require a target KG and can be applied

D. Garijo et al. (Eds.): ISWC 2025, LNCS 16140, pp. 235–252, 2026.
https://doi.org/10.1007/978-3-032-09527-5_13

to different ontologies as long as sufficient labeled data is available. However, manual annotation is costly, and thus such methods frequently suffer from the cold-start problem in a new domain.

Instead of relying on overlapping data or labeled data sources, which are often unavailable in new domains, we can use textual descriptions and a few examples from the target ontology to identify appropriate ontology classes and properties for each table column, similar to how humans approach the task. This semantic relatedness function can be learned through distant supervision using automatically generated labels from Wikipedia tables.

In this paper, we present a novel domain-independent approach for learning the semantic descriptions of tables without overlapping data with KGs. Our method is trained with distant supervision to estimate a score reflecting the semantic relatedness between a table column and an ontology class or property. For each column, we predict top K classes and properties. Then, the candidate classes and properties are combined to create the final semantic description. The empirical evaluation shows that our approach significantly outperforms strong baselines based on large language models. Our contribution is a novel method for the semantic modeling problem in domains with limited background knowledge and no labeled tables. Our method can be applied to domain ontologies that differ from those used during training, an important setting that has not been adequately addressed by the previous methods.

2 Example of Constructing a Semantic Description

In this section, we provide a real-world example to explain how information in the table and ontology can help create the semantic description of a table of soccer players, as shown in Fig. 1.

Scenario 1 (Using Table Data): We assume that each class and property in the ontology comes with a set of example values. For example, the class *association football position (Q4611891)* includes examples such as *goalkeeper (Q201330)* and *defender (Q336286)*, whereas the class *position (Q4164871)* contains examples like *president (Q30461)* and *senator (Q15686806)*. When examining the *position* column, we find that its values such as midfielder and goalkeeper are very similar to examples of the correct class *association football position (Q4611891)* and are different from examples of *position (Q4164871)*. These example values help disambiguate between candidate classes.

Scenario 2 (Using Table Metadata): For the *player* column, we do not find any cell values in the set of examples of the correct class *Q5 human*. Nonetheless, the column's header *player* strongly suggests that the column refers to people rather than places or organizations. This demonstrates how textual cues from the table header can guide semantic interpretation, even when the data values alone are insufficient.

player	height	position	team
Tom Aldred	1.88	Defender	Brisbane Roar
Shae Cahill	1.71	Midfielder	Brisbane Roar
Macklin Freke	1.93	Goalkeeper	Brisbane Roar

Fig. 1. Excerpt of a table of association football players with its semantic description on top. Green circle nodes are ontology classes; blue cells on the first table row represent columns. (Color figure online)

3 Approach

In this section, we explain our approach for the semantic modeling task, which is formally defined as follows:

Definition 1. ***(Semantic Modeling Problem).*** *Given a target ontology $\mathcal{O}$ and a table $s(c_1, c_2, ..., c_n)$, where each c_i is a column, predict the semantic description $sm(s)$ of the table. The semantic description $sm(s)$ is a graph whose nodes represent table columns, ontology classes, and literal values, and whose edges are ontology properties that capture semantic relationships between nodes.*

Figure 2 shows an overview of our approach, which consists of three main steps. First, we predict candidate classes and properties for each column in the input table. Second, we construct a candidate graph that encodes possible types and relationships among the columns. Finally, we apply a Steiner tree algorithm [20] to select a coherent subgraph that forms the final semantic description.

3.1 Column Concept Prediction

In this step, we identify candidate classes and properties for each table column. For conciseness, in this section, we refer to an ontology class or ontology property as a concept. For example, the column *position* in Fig. 1 has the top 3 concepts followed by their scores: *association football position (Q4611891)*: *0.87*, *position played on team (P413)*: *0.72*, and *position held (P39)*: *0.39*. If a column is an entity column, the predicted property will be the incoming relationship to the class associated with the column as shown in Fig. 1.

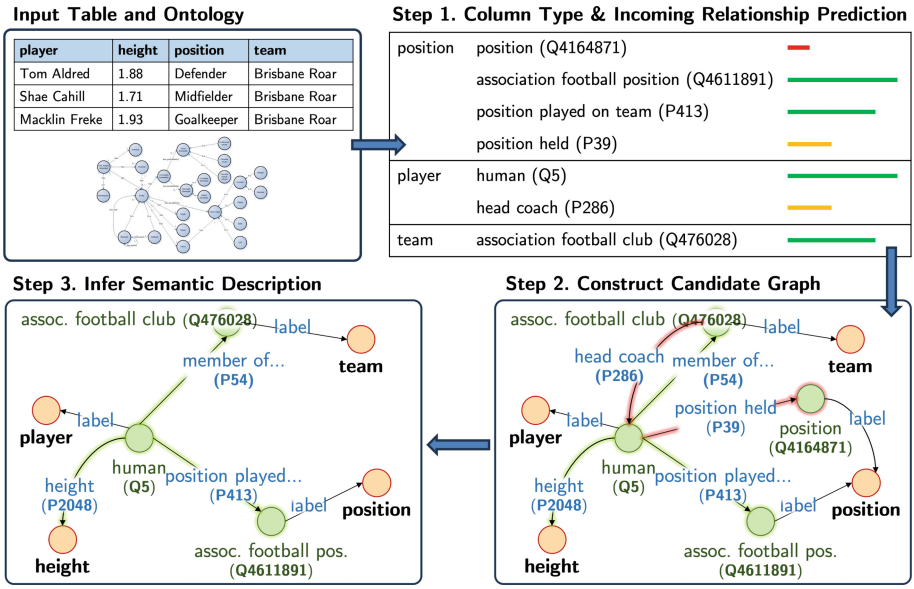

Fig. 2. Given an input table and a target ontology, we first predict candidate classes and incoming properties for each column. Next, we construct a candidate graph that encodes the possible types and relationships among the columns. Finally, we infer the semantic description by selecting a coherent subgraph that best represents the table's meaning.

Note that our goal differs from the traditional semantic labeling task [32]. The semantic labeling task predicts semantic types of a column, where a semantic type is a pair of a class and a property. In contrast, our goal is to predict a single class or a single property. Semantic labeling is more complex and may not be ideal for three reasons. First, we observe that many columns primarily express a property without an explicit associated class. For example, a column *capacity* may store the maximum capacity of stadiums, parks, restaurants, or vehicles. The correct class (e.g., *stadium*) is implicit and often requires a holistic view of the entire table to infer. This makes learning semantic types more difficult and can lead to ambiguity, especially when the same property is valid for many classes, thereby crowding out the top-K predictions with redundant combinations. The second reason is that predicting semantic types may leave out significant information that could be used in later processing steps. For example, the correct semantic type of a column *author* in Wikidata is ⟨*human (Q5)*, *rdfs:label*⟩, which fails to capture that the column contains authors (represented by the incoming *author (P50)* property). Finally, the number of possible class and property combinations can be enormous, as many properties are defined without strict domain constraints. For instance, the property *location (P276)* is broadly applicable and can be associated with a wide range of classes. As a

result, predicting class and property pairs can be computationally expensive and may not scale well to large ontologies.

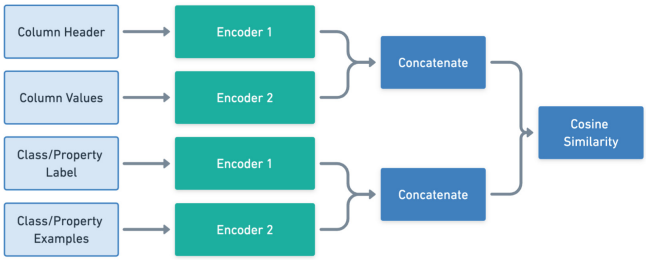

Fig. 3. Architecture of our neural network model to estimate a similarity score between a column and a concept.

To find the candidate concepts of a column, we learn to estimate the similarity score between the column and a concept based on the textual description and example values of the concept. This approach has two implications. First, we assume that a small set of example values is available for each class and property in the target ontology. In many cases, such examples can be retrieved automatically when classes or properties are linked (e.g., via *owl:sameAs*) to well-known concepts in public knowledge graphs. Second, we compute the similarity score for every concept in the ontology. While feasible for small or domain-specific ontologies, this becomes computationally impractical for large-scale ontologies like Wikidata, which include millions of classes and thousands of properties. One possible solution is to first classify the domain of a table, then cluster the concepts by domain or topic, and finally only compare concepts within the predicted domain. While promising, this direction is beyond the scope of the paper and is left for future work. For simplicity, in our evaluation, we limit the set of concepts to only the classes and properties that appear in the ground truth annotations.

Figure 3 shows the overall architecture of our neural network (NN) model for this task. First, it encodes a column name, column values, the name of a concept, and examples of the concept using encoders. Our encoders comprise a pre-trained transformer-based model[1] and a linear fully-connected layer. For data that are a list, such as column values or semantic label examples, we concatenate the data to create a sentence before feeding it to our encoders.

Our model is trained using a triplet loss [34]:

$$\mathcal{L}(\mathbf{x}, \mathbf{c}^+, \mathbf{c}^-) = \max(0, \text{dis}(\mathbf{x}, \mathbf{c}^+) - \text{dis}(\mathbf{x}, \mathbf{c}^-) + \epsilon)$$

where dis is the distance function between a column and a concept $\mathbf{c}$, $\mathbf{c}^+$ and $\mathbf{c}^-$ are the correct and incorrect concepts of the column, respectively, and the margin parameter ϵ is configured as the minimum offset between distances of similar vs

[1] We use BAAI/bge-m3 [4] in our experiment.

dissimilar pairs. In our experiment, we use $\text{dis}(a, b) = 1 - \text{cosine\_similarity}(a, b)$ and $\epsilon = 0.1$.

Our model is trained on the automatically labeled Wikipedia tables introduced by Vu et al. [41]. This dataset is constructed by linking table cells to Wikidata entities using Wikipedia hyperlinks, from which column types and relationships are inferred by matching the types and properties of associated entities across different columns. To improve labeling quality, a block list is applied to filter out examples where the column headers are incompatible with the inferred types (e.g., the column name is *soccer team*, but the labeled column type is *country (Q6256)*).

3.2 Semantic Description Prediction

Using the predicted concepts from the previous section, we perform two sub-tasks to construct the final semantic description of a table: column type prediction and column relationship prediction.

For column type prediction, we first identify entity columns using the entity column recognition algorithm introduced by [30]. This algorithm detects the types of individual cells (e.g., numbers, currency, dates), and considers a column to be an entity column if the majority of its cells are named entities. For each identified entity column, we assign the ontology class with the highest predicted score from the previous step as its correct type.

For column relationship prediction, we first construct a candidate graph that captures possible relationships among the columns in the table. Then, we employ a Steiner tree algorithm to select relationships that maximize the average likelihood as our final prediction.

Constructing Candidate Graphs. Algorithm 1 outlines the steps for constructing a candidate graph. It uses tuples of ⟨source class, property, qualifier, target class, frequency⟩, referred to as meta edges, as a guideline to discover relationships between classes and columns (lines 5 - 18). If the target ontology is a KG ontology, meta edges can be automatically extracted by grouping and counting property co-occurrences. Otherwise, they can be mined from Linked Open Data [38] or derived from domain and range constraints specified in the ontology.

The algorithm consists of two main steps. In the first step (lines 5 - 13), for each predicted property of a column, we use the meta edges to determine its source classes. Then, for each node associated with one of these source classes, we add a corresponding edge to the graph, but only if the domain and range constraints are satisfied. For example, if the predicted property is *author*, but the target column's predicted types are *company* and *publisher*, which do not appear in the range of *author*, then the property is rejected and no edge is added to the graph.

In the second step (lines 13 - 18), we iterate over each pair of source and target classes, identify the columns predicted to belong to those classes, and add edges between columns that contain properties capable of connecting the

Algorithm 1: CONSTRUCT CANDIDATE GRAPH

Input: the input table T,
columns' concepts: $\mathbb{C} = \{c_i : \{\text{class or property} : \text{score}, ...\}\}$,
indexes of entity columns,
list of meta edges: { ⟨ source class, property, qualifier, target class, frequency⟩, ... },
top K classes tkc,
top K properties tkp

Output: the candidate graph G

```
1  G ← empty graph
2  add columns in the table T as nodes to G
3  keep maximum top tkc classes and tkp properties for each column in ℂ
4  class2nodes ← a mapping from each ontology class to associated columns based
     on ℂ
   // add relationships between classes and columns
5  for column_index, col_classes, col_props ← ℂ do
6      for label, score ← col_props do
7          candidate_edges ← meta edges used label
8          for candidate_edge ← candidate_edges do
9              if domains and ranges of candidate_edge do not satisfy then
10                 continue
11             for node ← class2nodes do
12                 if node.column_index ≠ column_index then
13                     add candidate_edge between node and column_index to G
   // add relationships between classes
14 for source_class, source_nodes ← class2nodes do
15     for target_class, target_nodes ← class2nodes do
16         candidate_edges ← meta edges between source_class and target_class
17         for candidate_edge ← candidate_edges do
18             add candidate_edge between pairs of source_nodes and
                 target_nodes to G
19 return G
```

two classes. To limit the graph size and reduce potential noise, we consider only top-K predicted classes and top-K predicted properties for each column (K = 5 in our experiment) in the algorithm (line 3).

Finding the Steiner Tree. To identify the correct relationships from the candidate graph, we use a modified version of the BANKS algorithm [3], as adapted by [41], to compute a Steiner tree. The algorithm works by performing upward traversals from the terminal nodes (which represent table columns) until they converge at common roots. Candidate Steiner trees are then constructed by merging possible paths from each root to all terminal nodes. Since the number

of possible combinations grows exponentially, the algorithm uses beam search to find the tree with the highest average likelihood.

This algorithm requires assigning a likelihood score to each edge in the graph. We compute these scores as follows. Let

- $e = \langle u, v, p \rangle$ be an edge representing the relationship p between nodes u and v,
- f(v, c) be the predicted likelihood of the concept c for column v,
- sou_f(e) and tar_f(e) be the predicted likelihood of source and target class connected by e,
- $g(e)$ be the relative frequency of the property p being used to connect the source and target classes of e. The relative frequency is computed from the input meta edges.

We calculate the likelihood of e for different cases as follows. If e is a data property, then score$(e) = f(v, p) * \text{sou_f}(e)$. If e is an object property, then score$(e) = \text{sou_f}(e) * \text{tar_f}(e) * (f(v, p) * \alpha + g(e) * \beta)$, where α and β are hyper-parameters to balance the predicted likelihood by our model and the property popularity mined from the knowledge graph. Finally, if v is a statement node, then score$(e = \langle u, v, p \rangle) = \max_{t,p'}(\text{score}(e' = \langle v, t, p' \rangle))$.

4 Evaluation

Data. We evaluate our approach on the 250WT and T2Dv2 [33] datasets. We use the automatically labeled Wikipedia tables introduced by [41] to train our model. Both the training dataset and 250WT use the Wikidata ontology while T2Dv2 uses the DBpedia ontology. Table 1 shows statistics of these datasets. For the 250WT dataset, approximately 58% of classes and 66% of properties do not appear in the training dataset.

Metrics. We assess the performance on two tasks: column type annotation (CTA) and column relationship annotation (CPA) using the approximate precision, recall, and F1 metrics defined by the SemTab challenge [1]. These approximate metrics extend their exact version by giving partial credit when a prediction is close to the correct class or property, such as predicting a parent or child in the ontology hierarchy. Specifically, let $d(x)$ denote the shortest path from the predicted item x (a class or property) to the ground truth (GT). If x matches

Table 1. Statistics of datasets used in the evaluation

	train dataset	250WT	T2Dv2
#tables	18261	250	224
#classes	326	155	36
#properties	104	112	103

the ground truth exactly, or is its immediate parent or child, then $d(x) = 0$ or 1, respectively. The score for x is computed as follows: $\text{score}(x) = 0.8^{d(x)}$ if $d(x) \leq 5$ and x is a correct annotation or an ancestor of GT; $\text{score}(x) = 0.7^{d(x)}$ if $d(x) \leq 3$ and x is a descendant of GT; otherwise, $\text{score}(x) = 0$. For simplicity, we refer to these approximate scores as just precision, recall, or F_1.

Baselines. We developed several baselines to verify the effectiveness of our approach. The first baseline, named DSL-SM, uses DSL [32] to generate semantic types of columns of a table, then uses a frequent pattern mining method proposed by Taheriyan et al. [38] to combine the semantic types and create the semantic description of the table. The remaining baselines are based on large language models (LLM) that are trained and fine-tuned to follow textual instructions and to answer given questions. TableLlama [44] fine-tuned a Llama 2 (7B) model on 1.24 million tables for 11 tasks including CTA and CPA. We also compared with two recent LLM open-source models Llama 3.1-8B Instruct [13] and OLMo 2 [31]. For both models, we provide the table along with a list of ontology classes and properties as context, then ask the LLMs a series of questions to determine the types[2] of the columns and the relationships[3] between the columns.

Model Training. Our neural network model for column concept prediction is trained using AdamW [27] with a learning rate of 5e−5 for 5 epochs on an A5000 GPU. We also use 150 examples per target ontology class and property.

4.1 Overall Performance

Table 2 shows the performance of our method versus the baselines on the two datasets: 250WT and T2Dv2. On the 250WT dataset, our method outperforms the best-performing baseline, Llama 3.1-8B Instruct, on this dataset by 36.4% and 13.59% in F_1 scores on the CPA and CTA tasks, respectively. The baselines do not perform well on this hard dataset due to many classes and properties. Also, the precisions of LLM baselines are much lower than their recalls because they cannot predict if two columns have a relationship or not. Our gains in performance come from two sources. First, distant supervised training helps our method to better distinguish between candidate classes and properties. Second, finding the Steiner tree enables us to eliminate incorrect predictions to achieve high precision results.

All baselines achieve higher performance on the T2Dv2 dataset, as it is considerably simpler: each table is annotated with only one ontology class, and the target ontology (DBpedia) is less complex than the Wikidata ontology used

[2] Question's template: *What is the best class in the list above to describe the column {COLUMN}? Please keep your answer short with only the class or N/A if there is no suitable class. Also please wrap your answer with backticks. For example, {EXAMPLES}.*

[3] Question's template: *What is the best property in the list above to describe the relationship between the column {COLUMN1} and the column {COLUMN2}? Please keep your answer short with only the property or N/A if there is no suitable property. Also, please wrap your answer with backticks. For example, {EXAMPLE}.*

Table 2. Performance comparison between our method and the baselines on the CPA and CTA tasks. AP, AR, and AF_1 are macro-average approximate precision, recall, and F_1 in percentage, respectively

Dataset	Method	CPA			CTA		
		AP	AR	AF_1	AP	AR	AF_1
250WT	DSL-SM	19.53	22.03	19.71	27.76	27.97	27.37
	LLama 3.1- 8B Instruct	24.47	58.47	33.73	63.78	64.53	63.83
	OLMo 2	19.49	45.04	25.69	48.75	48.93	48.38
	TableLlama	23.06	29.79	21.20	41.00	35.65	35.73
	Our method	**76.12**	**67.38**	**70.13**	**77.36**	**78.35**	**77.42**
T2Dv2	DSL-SM	51.14	48.57	48.69	80.26	88.94	82.95
	LLama 3.1- 8B Instruct	56.99	55.95	56.01	70.53	67.85	67.85
	OLMo 2	63.29	60.99	61.50	91.22	89.44	89.44
	TableLlama	54.75	39.22	40.42	90.47	79.31	79.31
	Our method	**82.02**	**77.70**	**78.42**	**92.95**	**92.95**	**92.95**

in the 250WT dataset. Our method outperforms the strongest baseline on this dataset, OLMo 2, by 16.92% and 3.51% in F_1 score on the CPA and CTA tasks, respectively. These results demonstrate the effectiveness of our approach across different domains, despite using a significantly smaller model (0.5 billion versus 8 billion parameters). The performance gap of DSL-SM between the T2Dv2 and 250WT datasets further highlights the challenge of the traditional semantic labeling task in Wikidata, where the large number of classâĂŞproperty combinations greatly increases ambiguity and complexity.

Finally, TableLlama underperforms the other LLM baselines, even though it was fine-tuned for the CTA and CPA tasks. This is primarily because it tends to predict classes and properties seen in its training data, rather than selecting from those provided in the context.

4.2 Runtime

We evaluate the efficiency of our method by measuring the average runtime per table on the 250WT and T2Dv2 datasets. The experiment is conducted on a machine with an eight-core CPU and an A5000 GPU. Table 3 reports the results, comparing our method with two LLM baselines: Llama 3.1-8B Instruct and OLMo 2.

Our method is substantially faster than the LLM baselines on both datasets. On 250WT, our method processes a table in only 0.63 s on average, while Llama 3.1-8B Instruct and OLMo 2 require 32.93 and 35.74 s, respectively. Similarly, on T2Dv2, our method runs in just 0.28 s per table, compared to 8.28 s for Llama 3.1-8B Instruct and 6.88 s for OLMo 2. These results highlight the scalability and efficiency of our approach, making it suitable for large-scale table processing scenarios.

Table 3. Average runtime (in seconds) per table for the evaluated systems on the 250WT and T2Dv2 datasets

Method	250WT (sec.)	T2Dv2 (sec.)
Our method	0.63	0.28
Llama 3.1-8B Instruct	32.93	8.28
OLMo 2	35.74	6.88

4.3 Ablation Study

To assess the contribution of different input signals to overall performance, we conduct an ablation study by removing either the column header or the column values from the input. Table 4 reports the accuracy of each variant on the 250WT and T2Dv2 datasets.

The results reveal that both the column header and column values are important for accurate concept prediction. Removing either signal leads to a substantial drop in performance across both datasets. On the 250WT dataset, excluding the column header reduces the CTA F_1 score from 77.42% to 61.26%, while excluding the column values lowers it to 69.11%. A similar trend is observed for CPA. On T2Dv2, removing the header results in a 7.55-point drop in the CTA F_1 score and nearly a 25-point drop in the CPA F_1 score. Removing the values causes the CTA F_1 score to fall from 92.95% to 71.48%, and the CPA F_1 score to drop by 7.59 points.

An interesting observation is that, in many cases, the column values alone are sufficiently informative to identify the correct class or property. This is particularly evident in the T2Dv2 dataset, where using only the column values still yields a high CTA F_1 score of 85.40%. Upon examining examples from both datasets, we found that 50% of the columns have zero overlap with the provided examples of target concepts based on exact matching, and the average overlap is only around 13%. These results suggest that our approach effectively captures and leverages semantic similarity between the column values and the concept examples, even in the absence of direct matches. Futhermore, we found that the surrounding context of a table—such as the topic of the web page—can also play a crucial role. For instance, a table with just two columns, *year* and *title*, may have *title* values that resemble both books and films. While the column values alone may not clearly distinguish between the two, knowing that the table appears on a page about movies makes *film* the correct class. Incorporating such contextual signals is an interesting direction for future work.

5 Related Work

Prior research on the semantic modeling task falls into two broad categories: supervised methods designed for custom domain ontologies [8,9,11,36,37,40], and methods that leverage KGs [8,17,19,23,24,29,30,33,41,45].

Table 4. Performance of our method when (i) using only the column header, (ii) using only the column values, and (iii) using both (default). AP, AR, and AF_1 are macro-average approximate precision, recall, and F_1 in percentage, respectively

Dataset	Method	CPA			CTA		
		AP	AR	AF_1	AP	AR	AF_1
250WT	Our method	76.12	67.38	70.13	77.36	78.35	77.42
	- column headers	66.19	59.10	61.03	61.32	61.81	61.26
	- column values	63.54	59.36	60.39	69.16	69.84	69.11
T2Dv2	Our method	82.02	77.70	78.42	92.95	92.95	92.95
	- column headers	62.59	52.74	53.76	85.40	85.40	85.40
	- column values	74.48	70.83	71.03	71.48	71.48	71.48

Supervised approaches typically require two inputs: a target ontology and a set of tables annotated with their corresponding semantic descriptions. Taheriyan et al. [37] generate a semantic description by identifying a Steiner Tree that maximizes the coherence and frequency of classes and properties observed in the training data. Expanding on this idea, Vu et al. [40] propose a probabilistic graphical model (PGM) to estimate the likelihood of a semantic description and use it as a scoring function to guide the search for the most probable one. Feng et al. [11] enhance this line of work by incorporating a domain-specific KG in a post-processing step to further improve the accuracy of the predictions. More recent approaches based on transformer architectures, such as TURL [9], DODUO [36], and TorchicTab-Classification [8], frame the semantic modeling problem as a supervised classification task. In this setting, the model is trained to predict types for specific columns or relationships between given pairs of columns. However, this formulation is more constrained, as it assumes that the target columns are known in advance, an assumption that often does not hold in practice. Moreover, these methods require a huge amount of labeled training data. For example, TURL and DODUO are trained on a modified version of WikiGS [10], in which Wikipedia tables are labeled automatically with Freebase ontology. TorchicTab-Classification extends DODUO with a sub-table sampling strategy to work for large tables and is trained on SOTAB [22], a dataset constructed automatically from microdata embedded in websites. While supervised approaches offer flexibility in supporting arbitrary target ontologies, they suffer from the cold-start problem: the system requires a substantial number of labeled data sources to perform well. This limitation is even more pronounced for large ontologies, where extensive training data is needed to cover the wide range of classes and properties. In contrast, our approach does not require retraining to be applied to a new domain. Although it relies on example values for each ontology class or property, these examples can typically be obtained with minimal effort. For instance, we can reuse values from similar classes or properties in a knowledge graph, or prompt a large language model to generate them. Further-

more, performance on new domains can be further improved by simply adding relevant example values to the target table.

Methods that leverage existing knowledge in KG are typically unsupervised. However, some of them can incorporate labeled data to fine-tune their parameters. These approaches generally follow a common pipeline: they begin by linking table cells to KG entities (Entity Linking, or EL), and then use the properties associated with those entities to infer column types (CTA) and column relationships (CPA). Limaye et al. [24] and Mulwad et al. [29] are among the first works to solve the three tasks (EL, CTA, and CPA) jointly using probabilistic graphical models. However, their methods are limited to entity columns and do not accommodate columns containing literal values. Subsequent research shifted toward iterative approaches and expanded the scope to include literal columns. Ritze et al. [33] proposed an iterative method that begins by identifying the subject (entity) column of a table, followed by generating candidate relationships between the subject and the remaining columns. The method then refines both the candidate entities and relationships over multiple iterations until convergence. Zhang et al. [45] further improved this process by refining entity linking outputs to be consistent with both the predicted column types and the overall table domain, which is estimated using a bag-of-words approach. More recent systems that achieved top performance in the SemTab challenges [1,7,15,21], such as MTab [30], DAGOBAH [19], and others KGCode-Tab [23], LinkingPark [6], BBW [35], TorchicTab-Heuristic [8], and SemTex [17] also follow the iterative paradigm. These systems introduce improvements at various stages of the pipeline, such as more accurate candidate entity retrieval and enhanced scoring mechanisms for ranking annotations. GRAMS [42] extends the semantic modeling task to the full setting, supporting the prediction of n-ary relationships and missing context values in Wikipedia tables. Specifically, they construct a candidate graph of potential relationships and use a Probabilistic Soft Logic [2] to rank the edges and select the most likely subgraph that represents the table's semantics. GRAMS+ [41] further generalizes this framework to handle non-Wikipedia tables by automatically generating labeled data from Wikipedia to train models that score candidate entities and candidate relationships. Compared to our approach, methods that rely on knowledge graphs are constrained to tables that contain overlapping data with the target KG and can only map to the KG's ontology. In contrast, our method does not depend on such overlap and can be applied to a wide range of domains and ontologies without retraining.

Finally, there is some previous work on table understanding that has different or restricted settings compared to the semantic modeling task in this paper [25]. For example, DSL [32], ColNet [5], JellyFish [43], Sherlock [18] focus on only annotating the column types in the tables. Tab2KG [12]'s core idea is to address the semantic labeling task similar to DSL using features such as histograms, average text length extracted from data in a target knowledge graph. However, after mapping columns to pairs of ontology classes and properties, Tab2KG constructs the final semantic description by assigning object properties to class pairs without resolving ambiguities (i.e., treating all candidate relation-

ships equally without disambiguation). Luzuriaga et al. [28] focus on generating triples of relationships between columns for Wikipedia tables. The Metadata-to-KG track in the SemTab 2024 challenge [16] focuses on mapping a given column to an ontology property using only the table header. Adwan [39], the winning system in this track, employs an LLM with a retrieval-augmented generation (RAG) approach to first retrieve candidate properties and then rerank them. The dataset used in the first round is a subset of the T2Dv2 dataset, and the system achieves an accuracy of 62%. The authors also present an alternative method that achieves higher accuracy (75%) by retrieving properties from the LLM parametric memory rather than from the context. This is motivated by findings that top-performing LLMs are often pretrained on DBpedia data [26].

6 Conclusion

In this paper, we presented a novel, domain-independent method for semantic table interpretation that does not rely on overlapping data with a knowledge graph. Using distant supervision, our method learns to identify the top-K most relevant classes and properties for each table column and constructs semantic descriptions based on these candidates. Our evaluation demonstrates that the approach achieves strong performance across different ontologies, including those in unseen domains. This makes the method especially effective in scenarios where labeled data is scarce or unavailable.

Our approach is also particularly well-suited for semi-automated systems for publishing tables to knowledge graphs, such as Karma [14]. Since our method produces the top-K candidate classes and properties for each column, it can assist users in correcting errors introduced by the automated semantic modeling process. Moreover, because our method does not require retraining to incorporate newly annotated tables, it enables more efficient and incremental updates, an important feature for interactive systems where retraining may be time-consuming.

7 Future Work

There are several promising directions for extending this work. First, while our method computes similarity scores between a table column and every class and property in the target ontology, this becomes computationally expensive for large ontologies such as Wikidata, which contains millions of entities and thousands of relations. A potential solution is to first classify the domain of a table and then identify the subset of ontology concepts that are relevant to that domain based on their usage in the knowledge graph. This strategy could significantly reduce the number of concepts that need to be considered, making the approach more scalable to large and heterogeneous ontologies.

Second, we believe that the performance of the column concept prediction component (Sect. 3.1) can be further improved with models that better leverage the surrounding context of the table. One approach we experimented with

involves fine-tuning a Llama model that is given the full table, a candidate concept (including its textual description and example values), and prompted with a yes/no question asking whether the concept correctly describes the given column. The similarity score is then computed by applying a softmax over the logits corresponding to the *Yes* and *No* responses. While preliminary results showed lower overall performance compared to our current model, this LLM-based approach achieved a notable 8.43% improvement in CPA F_1 score (86.85%) on the T2Dv2 dataset. However, the main drawback is its high inference cost, due to the large number of combinations between columns and candidate concepts. Specifically, prediction took 93 h (22 minutes per table) for the 250WT dataset and 26 h (7 minutes per table) for the T2Dv2 dataset. Finding efficient strategies to reduce the cost would be an important step toward making this approach practical at scale.

Finally, many real-world tables contain auxiliary or uninformative columns that may not correspond to any class or property in the target ontology. Introducing an "unknown" concept, augmented with previously seen column headers and cell values, could help our concept prediction component identify and filter out such columns. This capability would not only improve the robustness of semantic labeling but also enhance overall performance by reducing noise during prediction.

Acknowledgments. This material is based upon works supported by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112390132 and Contract No. 140D0423C0093. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA or its Contracting Agent, the U.S. Department of the Interior.

The first author is grateful for the support of the Vietnam Education Foundation (VEF) Fellowship, which provided funding during the early stages of this research.

Supplemental Material Statement. The source code for our approach is publicly available at https://github.com/binh-vu/iswc25.

References

1. Abdelmageed, N., et al.: Results of semtab 2022. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2022). CEUR Workshop Proceedings, vol. 3320. CEUR (2022). https://openaccess.city.ac.uk/id/eprint/29744/, 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)
2. Bach, S.H., Broecheler, M., Huang, B., Getoor, L.: Hinge-loss markov random fields and probabilistic soft logic. J. Mach. Learn. Res. **18**(109), 1–67 (2017)
3. Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., Sudarshan, S.: Keyword searching and browsing in databases using BANKS. In: Proceedings 18th International Conference on Data Engineering, pp. 431–440 (2002). https://doi.org/10.1109/ICDE.2002.994756

4. Chen, J., Xiao, S., Zhang, P., Luo, K., Lian, D., Liu, Z.: Bge m3-embedding: multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation (2024)
5. Chen, J., Jiménez-Ruiz, E., Horrocks, I., Sutton, C.: ColNet: embedding the semantics of web tables for column type prediction. AAAI **33**(01), 29–36 (2019)
6. Chen, S., et al.: LinkingPark: an integrated approach for semantic table interpretation. https://ceur-ws.org/Vol-2775/paper7.pdf?ref=https://githubhelp.com, Accessed 6 Oct 2023
7. Cutrona, V., et al.: Results of semtab 2021. In: 20th International Semantic Web Conference. vol. 3103, pp. 1–12. CEUR Workshop Proceedings (2022). https://openaccess.city.ac.uk/id/eprint/28056/, copyright 2021 for the individual papers by the papers' authors. This volume and its papers are published under the Creative Commons License Attribution 4.0 International (CC BY 4.0)
8. Dasoulas, I., Yang, D., Duan, X., Dimou, A.: TorchicTab: semantic table annotation with wikidata and language models. In: CEUR Workshop Proceedings, pp. 21–37. CEUR Workshop Proceedings (2023)
9. Deng, X., Sun, H., Lees, A., Wu, Y., Yu, C.: TURL: table understanding through representation learning (2020)
10. Efthymiou, V., Hassanzadeh, O., Rodriguez-Muro, M., Christophides, V.: Matching web tables with knowledge base entities: from entity lookups to entity embeddings. In: The Semantic Web – ISWC 2017, pp. 260–277. Springer International Publishing (2017). https://doi.org/10.1007/978-3-319-68288-4_16
11. Feng, Z.W., et al.: Automatic semantic modeling for structural data source with the prior knowledge from knowledge graph (2021)
12. Gottschalk, S., Demidova, E.: Tab2kg: semantic table interpretation with lightweight semantic profiles. Semantic Web **13**(3), 571–597 (2022)
13. Grattafiori, A., et al.: The llama 3 herd of models. arXiv preprint arXiv:2407.21783 (2024)
14. Gupta, S., Szekely, P., Knoblock, C.A., Goel, A., Taheriyan, M., Muslea, M.: Karma: a system for mapping structured sources into the semantic web. In: Extended Semantic Web Conference, pp. 430–434. Springer (2012)
15. Hassanzadeh, O., et al.: Results of semtab 2023. In: CEUR Workshop Proceedings. vol. 3557, pp. 1–14 (2023)
16. Hassanzadeh, O., et al.: Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching co-located with the 23rd International Semantic Web Conference (ISWC 2024), Baltimore, USA, November 13, 2024, CEUR Workshop Proceedings, vol. 3889. CEUR-WS.org (2025). https://ceur-ws.org/Vol-3889
17. Henriksen, E.G., et al.: Semtex: A hybrid approach for semantic table interpretation (2023)
18. Hulsebos, M., Hu, K., et al.: Sherlock: a deep learning approach to semantic data type detection. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1500–1508. KDD '19, Association for Computing Machinery, New York (2019)
19. Huynh, V.P., Chabot, Y., Labbé, T., Liu, J., Troncy, R.: From heuristics to language models: a journey through the universe of semantic table interpretation with DAGOBAH. Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) (2022)
20. Hwang, F.K., Richards, D.S.: Steiner tree problems. Networks **22**(1), 55–89 (1992)

21. Jimenez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., Srinivas, K., Cutrona, V.: Results of semtab 2020. CEUR Workshop Proc. **2775**, 1–8 (2020). http://ceur-ws.org/Vol-2775/, copyright 2020 for this paper by its authors
22. Korini, K., Peeters, R., Bizer, C.: Sotab: the wdc schema. org table annotation benchmark. In: CEUR Workshop Proceedings. vol. 3320, pp. 14–19. RWTH Aachen (2022)
23. Li, X., et al.: KGCODE-Tab results for SemTab 2022. https://ceur-ws.org/Vol-3320/paper5.pdf, Accessed 6 Oct 2023
24. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. In: Proceedings of the VLDB Endowment. vol. 3, pp. 1338–1347. VLDB Endowment (2010)
25. Liu, J., Chabot, Y., Troncy, R., Huynh, V.P., Labbé, T., Monnin, P.: From tabular data to knowledge graphs: a survey of semantic table interpretation tasks and methods. J. Web Semantics **76**, 100761 (2023)
26. Lo, P.C., Tsai, Y.H., Lim, E.P., Hwang, S.Y.: On exploring the reasoning capability of large language models with knowledge graphs. arXiv preprint arXiv:2312.00353 (2023)
27. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019). https://openreview.net/forum?id=Bkg6RiCqY7
28. Luzuriaga, J., Munoz, E., Rosales-Mendez, H., Hogan, A.: Merging web tables for relation extraction with knowledge graphs. IEEE Trans. Knowl. Data Eng., 1–1 (2021)
29. Mulwad, V., Finin, T., Joshi, A.: Semantic Message Passing for Generating Linked Data from Tables. In: Alani, H., et al., (eds.) ISWC 2013. LNCS, vol. 8218, pp. 363–378. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41335-3_23
30. Nguyen, P., Yamada, I., Kertkeidkachorn, N., Ichise, R., Takeda, H.: SemTab 2021: tabular data annotation with MTab tool. http://ceur-ws.org/Vol-3103/paper8.pdf, Accessed 6 Oct 2023
31. OLMo, T., et al.: 2 olmo 2 furious. arXiv preprint arXiv:2501.00656 (2024)
32. Pham, M., Alse, S., Knoblock, C.A., Szekely, P.: Semantic labeling: a domain-independent approach. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) ISWC 2016. LNCS, vol. 9981, pp. 446–462. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46523-4_27
33. Ritze, D., Lehmberg, O., Bizer, C.: Matching HTML tables to DBpedia. In: Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics, pp. 1–6. No. Article 10 in WIMS '15, Association for Computing Machinery, New York (2015)
34. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: a unified embedding for face recognition and clustering. In: Proceedings/CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015). https://doi.org/10.1109/CVPR.2015.7298682, https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Schroff_FaceNet_A_Unified_2015_CVPR_paper.html
35. Shigapov, R., Zumstein, P., Kamlah, J., Oberlander, L., Mechnich, J., Schumm, I.: bbw: matching CSV to wikidata via meta-lookup. https://madoc.bib.uni-mannheim.de/57386/3/paper2.pdf, Accessed 6 Oct 2023

36. Suhara, Y., et al.: Annotating columns with pre-trained language models. In: Proceedings of the 2022 International Conference on Management of Data, New York. ACM (2022)
37. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: Learning the semantics of structured data sources. J. Web Semantics **37–38**, 152–169 (2016)
38. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: Leveraging linked data to discover semantic relations within data sources. In: Lecture Notes in Computer Science, pp. 549–565. Lecture notes in computer science, Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-46523-4_33
39. Vandemoortele, N., Steenwinckel, B., Hoecke, S., Ongenae, F.: Scalable table-to-knowledge graph matching from metadata using LLMS (2024)
40. Vu, B., Knoblock, C., Pujara, J.: Learning semantic models of data sources using probabilistic graphical models. In: The World Wide Web Conference, pp. 1944–1953. WWW '19, Association for Computing Machinery, New York (2019)
41. Vu, B., Knoblock, C.A., Shbita, B., Lin, F.: Exploiting distant supervision to learn semantic descriptions of tables with overlapping data. In: International Semantic Web Conference, pp. 116–134. Springer (2024)
42. Vu, B., Knoblock, C.A., Szekely, P., Pham, M., Pujara, J.: A Graph-Based Approach for Inferring Semantic Descriptions of Wikipedia Tables. In: Hotho, A., et al., (eds.) ISWC 2021. LNCS, vol. 12922, pp. 304–320. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88361-4_18
43. Zhang, H., Dong, Y., Xiao, C., Oyamada, M.: Jellyfish: a large language model for data preprocessing. arXiv preprint arXiv:2312.01678 (2023)
44. Zhang, T., Yue, X., Li, Y., Sun, H.: Tablellama: towards open large generalist models for tables. arXiv preprint arXiv:2311.09206 (2023)
45. Zhang, Z.: Effective and efficient semantic table interpretation using TableMiner+. Semant. Web **8**(6), 921–957 (2017)