Exploiting Distant Supervision to Learn Semantic Descriptions of Tables with Overlapping Data

Binh Vu[1], Craig A. Knoblock[1], Basel Shbita[1], and Fandel Lin[1]

USC Information Sciences Institute, Marina del Rey CA 90292, USA
{binhvu,knoblock}@isi.edu and {shbita,fandel.lin}@usc.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 that exploits the matches between entities in a knowledge graph and the table data does not perform well for tables with noisy or ambiguous data. A key reason for this poor performance is the limited amount of labeled data to train these methods. To address this problem, we propose a novel distant supervision approach that leverages existing Wikipedia tables and hyperlinks to automatically label tables with their semantic descriptions. Then, we use the labeled dataset to train neural network models to predict the semantic description of a new table. Our empirical evaluation shows that using the automatically labeled dataset provides approximately 5% improvement in column type prediction and 4.5% improvement in column relationship prediction in F1 scores over the state-of-the-art on a large set of real-world tables.

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

1 Introduction

The task of building a semantic description of a table, or semantic modeling, is to annotate the types and relationships of columns using classes and predicates of an ontology. The resulting annotation, called a semantic description, can be used both for data discovery and for publishing data to a knowledge graph (KG).

Since creating semantic descriptions requires significant effort and expertise, there is much research on automating this problem. In general, they can be classified into two groups. The first group is supervised methods trained on a set of known semantic descriptions with given domain ontologies [35, 36, 13]. Because the manually labeled dataset is expensive to obtain, these methods do not work well for new or large ontologies. The second group exploits the background knowledge in KGs to predict the semantic description [25, 28, 25, 28, 32, 39, 29, 20, 24]. First, they link table cells to entities in KGs. Then, they match the entities' property values with other table cells to find the semantic description. These approaches work for large ontologies and generally do not

need to be retrained when the ontology is updated. They also tend to have only a few parameters to learn; hence, they typically need little to no training data.

However, approaches in the second group tend to perform poorly on tables with lots of ambiguity such as having similar candidate entities or few matches. Due to the limited amount of training data, it is challenging for these methods to learn the optimal parameters to combine the entity linking and data matching results to resolve the ambiguity. Meanwhile, Wikipedia has millions of tables containing hyperlinks that can be automatically converted to entities in a KG such as Wikidata. This poses an opportunity to leverage this data to learn the semantics of tables.

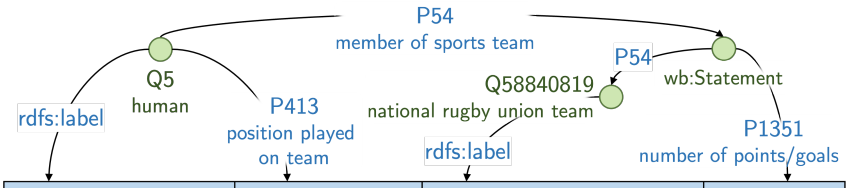
In this paper, we present a novel approach that exploits distant supervision for semantic modeling. Specifically, we generate labeled datasets from Wikipedia tables to train two neural networks (NN) to predict the likelihood of candidate entities and column relationships. Then, similar to methods in the second group, we perform entity linking and predict column types and relationships using the two NNs. In our empirical evaluation, training the NN models with distant supervision provides approximately 5% improvement in column type prediction and 4.5% improvement in column relationship prediction in F1 scores over the state-of-the-art.

The contribution of this paper is a novel method of using distant supervision for the semantic modeling problem. Our solution provides an algorithm to create the semantic description of a given table that is more robust to noise and ambiguity in the table data than the previous state-of-the-art. We also demonstrate that the automatically labeled dataset, although it may be noisy, can be useful to help train machine learning models to understand the semantics of tables.

2 Motivating Example

In this section, we provide an example to explain the problem of inferring a semantic description of a table using the background knowledge found in a KG, such as Wikidata. We want to map a table of players of national rugby teams to Wikidata's ontology. In Wikidata, each item (class/property) is uniquely identified by a letter followed with a number (e.g., `Q5` or `P54`). To make it easier to read, we occasionally add the item label after the identifier such as `Q5 human`.

Figure 1 shows an excerpt of the table with its semantic description on top. The semantic description is expressed as a graph because there can be n-ary relations between columns in a table. Each node in the graph represents a column, an ontology class, or a literal (e.g., number, text, or date). Each edge presents an ontology predicate encoding relationships between the two nodes. For example, the edge `Q5 human` $\xrightarrow{\texttt{rdfs:label}}$ `Player` specifies that the `Player` column contains people's names. The edge (`Q5 human` $\xrightarrow{\texttt{P54 members of sports teams}}$) shows that they are members of some sports teams. Note that the table also has the number of points that players scored for their teams. This is a ternary relationship and is expressed using an intermediate node `wb:Statement` called statement node. To indicate what the statement is mainly about, we use an outgoing edge (e.g.,



Player	Position	Team	Points
Dan Carter	Fly-half	New Zealand	1598
Ronan O'Gara	Fly-half	Ireland	1083
Percy Montgomery	Fullback	South Africa	893
Chris Paterson	Fullback	Scotland	809

Fig. 1. Excerpt of a table of players of national rugby teams with its semantic description on top. Green circle nodes are ontology classes; blue cells on the first table row represent columns. The node `wb:Statement`, named statement node, is used to represent a ternary relationship between the players, their teams, and the number of points scored for the teams.

`P54`) that has the same label as the statement's incoming edge. For instance, `Q5 human` $\xrightarrow{\texttt{P54}}$ `wb:Statement` $\xrightarrow{\texttt{P54}}$ `Team` reveals that the players' teams can be found in the `Team` column. We refer to the edge `P54` as the statement property. The other outgoing edges are statement qualifiers providing additional information to the statement (e.g., `wb:Statement` $\xrightarrow{\texttt{P1351 number of points}}$ `Points`).

The first step in methods that exploit a KG as background knowledge is to detect linkable cells and then retrieve candidate entities associated with the cells. For example, `Dan Carter` can link to `Dan Carter (politician)` or `Dan Carter (rugby player)`. Similarly, `New Zealand` can link to `New Zealand (country)`, `New Zealand national football team`, or `New Zealand national rugby team`. Then, each property's value of the candidate entities of a cell is matched with other cells in the table to identify potential relationships between the two cells. For example, the value of the property `P27 country of citizenship` of `Dan Carter (rugby player)` is `New Zealand (country)` suggesting that `P27` can be the relationship between the columns `Player` and `Team`.

The ambiguity arises as the correct entity of the cell `New Zealand` is the rugby team, but the candidate entity `New Zealand (country)` has a label that matches exactly with the cell's value. In addition, property `P27 country of citizenship` is found in more rows than the correct property `P54` making it even more uncertain. Fortunately, the surrounding context such as the column header `Team` tells us that entities in this column should not be countries. Also, we find more members of rugby teams in the table than members of football teams. Thus, rugby teams should have higher likelihood, which are the correct entities. Without a large training dataset, learning to combine all of these signals can be challenging with large ontologies and noisy tables. Therefore, we turn our attention to using the hundreds of thousands of Wikipedia tables with hyperlinks to other articles. These links enable us to infer the ground-truth entities and parts of the semantic descriptions with decent accuracy. Then, we use the inferred

labeled data to train a context-aware table entity linking method, and a neural network model to predict the semantic descriptions.

3 Learning the Semantics of Tables

Problem Definition: In this paper, we focus on the problem of finding columns' types and their (n-ary) relationships. Given a plain[1] relational table s and target ontology $\mathcal{O}$ of a knowledge graph, we want to find the semantic description $sm(s)$ of table s using classes and predicates in $\mathcal{O}$.

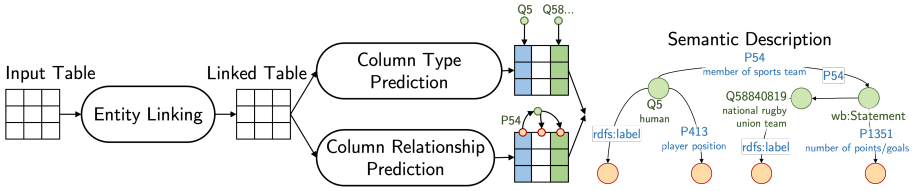


Fig. 2. Overall approach

Overall Approach: Figure 2 shows the overall approach. It starts by finding KG entities that are mentioned in a table (entity linking). Then, we use a neural network (NN) to compute the scores of candidate entities of each table cell. The NN model is trained with distant supervision. Using the discovered candidates and their scores, we perform two tasks: predict column types (CTA) and column relationships (CPA). The results of CTA and CPA are combined to get the final semantic description. In our experiment, finding column relationships is much more difficult than finding the column types. To combine all potential signals for the CPA task, we use another NN model to estimate the likelihood of possible relationships. This model is also trained using distant supervision.

3.1 Creating Labeled Dataset from Wikipedia Tables

To automatically annotate Wikipedia tables, we leverage the hyperlinks inside the tables to find corresponding Wikidata entities and predict columns' relationships based on the linked entities. However, it's a non-trivial task to label all Wikipedia tables with high accuracy. There are several challenges such as detecting table layout and orientation, identifying location of headers and content, or context-inconsistent hyperlinks. An example of context-inconsistent hyperlinks is a column named `city`, but the links in the column are to `airport`.

For simplicity, we focus on relational tables for which it is relatively easy to automatically generate labeled data. We define the following conditions to identify an easy-to-label table: (1) has a minimum of 10 rows, (2) has a maximum of one hyperlink per cell, (3) has a column with at least 70% of its cells containing hyperlinks, and (4) more than 80% of the links have corresponding

[1] A plain table does not contain any markup such as hyperlinks

entities in the KG. The conditions are derived heuristically based on the quality of the generated labels of a small set of randomly selected Wikipedia tables used for development purposes. To remove context-inconsistent hyperlinks, we first automatically assign a type to each column based on the most common type of its entities. Then, we employ a blocklist to remove all links in a column if the column header is incompatible with the predicted column types. We create the blocklist by manually reviewing normalized column headers[2] that appear in multiple predicted types and labeling the incompatible types. The number of headers to review is approximately 230 headers.

To generate labeled data for entity linking, we use the existing hyperlinks as positive examples. The negative examples are other candidate entities retrieved using the method described in Section 3.2. To generate labeled data for column relationship prediction, we first apply our method described in Section 3.4 to create a graph containing candidate relationships, then remove the relationships that occur in less than 50% of the rows or have more than 10% of the rows with different data than in KGs. After that, the remaining relationships are grouped by the source and target columns or literal values; only the relationships with the highest matching frequencies are used as the ground truth for the column relationship prediction. The generated candidate relationships that are not in the ground truth are considered as negative examples. The numbers used in the aforementioned steps are also chosen heuristically based on the small set of Wikipedia tables.

3.2 Entity Linking in Tables

The first step in our method is to link table cells to entities in the KG. Following typical entity linking (EL) systems, our EL approach consists of three main steps: (1) detect the entity columns, which are the cells that will be linked; (2) retrieve candidate entities for each cell; and (3) compute the candidates' likelihood.

To detect entity columns, we use the same heuristic as in [29]. Specifically, if the majority of the cells of a column are classified as text (e.g., using Spacy and regex), the column will be linked. This heuristic yields a high recall but low precision (about 0.75 to 0.8 on the evaluation datasets). Next, we generate candidate entities for each cell in the entity columns by combining search results of several approaches: public search API (e.g., Wikidata API), keyword search using ElasticSearch, and fuzzy matching of entity names [14]. Finally, we use a neural network that takes a cell, the surrounding context (e.g., column header), and a candidate entity, then predicts the likelihood of the candidate entity.

Our candidate entity scoring model is a two-hidden-layer perceptron with RELU activations. It is trained using the auto-label dataset described in Section 3.1 with the following groups of features.

Surface Features consist of four different metrics to measure the similarities between a mention and candidate names: Levenshtein, Jaro-Winkler, Monge Elkan, and Generic Jaccard.

[2] We normalize a header by masking numbers, removing special characters, etc.

Entity-context Similarity Features capture the coherence between a candidate and the surrounding context of a mention. We have two context similarity features: the weighted dot product of the embeddings of the column header and the candidate's description, and the number of cells matched with the candidate's property divided by a large constant representing the maximum number of columns in a table (e.g., 20) for rescaling. The embeddings are computed from a Sentence Transformer model [31][3], and the weights of embedding dimensions are learnable parameters.

Entity Prior Features bias the predictions toward popular entities. Currently, we use the normalized log page rank of a candidate as the prior feature. The normalized log page rank of an entity e is calculated as follows:

$$\frac{\log(\text{pagerank}(e)) - \min_{e' \in \mathcal{E}} \log(\text{pagerank}(e'))}{\max_{e' \in \mathcal{E}} \log(\text{pagerank}(e')) - \min_{e' \in \mathcal{E}} \log(\text{pagerank}(e'))}$$

where $\mathcal{E}$ is the set of entities in KG, pagerank(e) is the pagerank of an entity e.

As real-world tables sometimes do not have headers, we train another version of our entity linking model on the same auto-label dataset in which the headers have been removed. At the testing time, if a column header is empty, we use the model trained without headers. Otherwise, we use the model trained with headers. We find that this strategy works better than training a single model on the combined datasets.

3.3 Column Type Prediction

Algorithm 1 describes our column type prediction method. This greedy algorithm first selects the type with the highest score from the set of types directly found in the candidate entities (lines 1 - 3) of a column. The score of a type is computed by summing the maximum likelihood of the candidate entities of the type for each cell (line 2) and divided by the number of rows (line 3). Next, the algorithm iteratively refines the prediction by replacing it with an ancestor type within d distance of the directed types if the score difference is larger than a specific threshold δ (lines 4 - 10). d is increased by one after each iteration up to max_distance, which is a parameter of the algorithm. The threshold ($\delta = 0.1$) and maximum distance (max_distance $= 2$) are chosen empirically.

To illustrate the algorithm, we use an example of a column named `university` containing ten cells. The annotated type of the column is `Q3918 university`. We have eight cells linked to public universities and two cells linked to private universities. First, the algorithm finds two candidate types `Q875538 public university` and `Q902104 private university` from the entities in the column (line 1), which the likelihoods are 0.8 and 0.2, respectively (line 2 - 3). Thus, it assigns the best type to be `Q875538` (line 3). Then, it considers ancestors of the discovered types within 1-hop and discovers `Q3918 university` (line 5). The likelihood of the new type is 1.0 (line 6 - 7). Since it is greater than $0.8 + \delta$,

[3] We use the pretrained all-mpnet-base-v2 model.

Algorithm 1: Column Type Prediction

Input: A target column C, candidate entities of each cell in C, a threshold δ, and a maximum searching distance max_distance
Output: predicted column type and its score

```
di_types ← direct types of candidate entities in C
type2cells ← mapping from a type in di_types to the maximum likelihood of
  candidate entities of the type (no inheritance) for each cell in C
best_type, best_avg_likelihood ← the type with the highest average likelihood
  from type2cells
for d ← 1..max_distance do
    extend_types ← di_types and ancestors of di_types within d-hop
    extend_type2cells ← mapping from a type in extend_types to the
      maximum likelihood of candidate entities of the type (with inheritance)
      for each cell in C
    new_type, new_avg_likelihood ← the type with the highest average
      likelihood from extend_type2cells
    if new_avg_likelihood ≥ best_avg_likelihood + δ then
        best_type ← new_type
        best_avg_likelihood ← new_avg_likelihood
return best_type, best_avg_likelihood
```

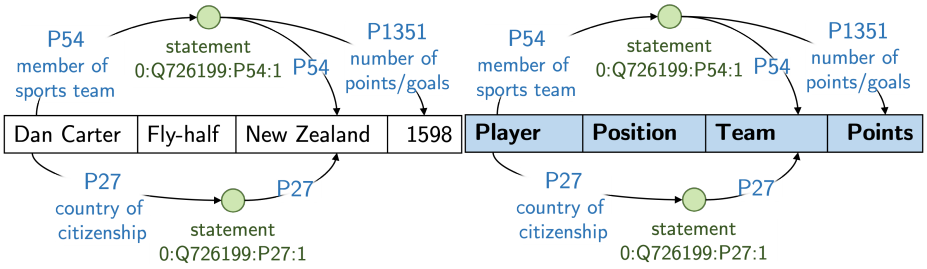
the best type is reassigned to `Q3918`. In the second iteration, it searches ancestors within 2-hop distance, and finds `Q2385804 educational institution`, of which likelihood is also 1.0. However, the algorithm will not reassign the best type as the score difference is now less than δ. The iteration ends and the algorithm returns `Q3918`, which is the correct type.

3.4 Column Relationship Prediction

Our column relationship prediction consists of three steps. First, we generate a candidate graph containing potential relationships between columns. Then, we use a classifier to predict the likelihood of each link in the graph. Lastly, we employ the Steiner Tree algorithm [35] to select the links that maximize the average likelihood as our final prediction.

The candidate graph is created using a modified version of the algorithm in [37]. The general idea of the algorithm is to build a data graph, containing relationships between cells in a table. Then, we group the relationships between the cells from the same pairs of columns to obtain the relationships at the column level. Different from the original version, our modified algorithm treats each row of the table independently. Therefore, when building the data graph, we can process the table in parallel while not having hubs with many connected edges.

To discover the relationships between cells, for each row, we iterate through each candidate entity of a cell and match the property value of the candidate with other cells in the same row. When we find a match, we add a statement node and link from the cell to the matched cell through the statement node. The statement node is uniquely identified by the row number as well as the original statement id in Wikidata. This enables n-ary relationships to be constructed automatically during the matching process. For example, in the first row of the table in Figure 1,



(a) The data graph after processing the first row of the table

(b) Excerpt of the candidate graph built from the data graph

Fig. 3. Excerpt of a data graph and a candidate graph containing relationships discovered in the table in Figure 1

assuming that we have the following candidate entity `Q726199` for `Dan Carter`, and ⟨`Q664`, `Q55801`⟩ for New Zealand. We discover the following relationships: (`Q726199` $\xrightarrow{\texttt{P54 members of sports teams}}$ `statement:0:Q726199:P54:1` $\xrightarrow{\texttt{P54}}$ `Q55801`), (`Q726199` $\xrightarrow{\texttt{P54}}$ `statement:0:Q726199:P54:1` $\xrightarrow{\texttt{P1351}}$ `1598`) , and (`Q726199` $\xrightarrow{\texttt{P27}}$ `statement:0:Q726199:P27:1` $\xrightarrow{\texttt{P27 country of citizenship}}$ `Q664`). The data graph after processing the first row is shown in Figure 3a.

From the data graph, the candidate graph is created by grouping links of nodes between the same pair of columns. The grouping is performed in two phases: links representing statement properties are grouped before links representing statement qualifiers. For example, in Figure 3a, we will first group links ($\xrightarrow{\texttt{P54}}$ `statement:*:*:P54` $\xrightarrow{\texttt{P54}}$) between cells of columns `Player` and `Team`. Then, we add the qualifiers (`statement:*:*:P54` $\xrightarrow{\texttt{P1351}}$) to the grouped statement node. An excerpt of the candidate graph is shown in Figure 3b.

The classifier employed to predict the likelihood of links is also a two-hidden-layer perceptron with RELU activations. It is trained on the auto-label dataset (Section 3.1) to take features extracted from a single link and classify whether it is correct. The features include the relative frequency of discovering the link from top K entities, the average link likelihood, the relative frequency of finding contradicting information between the table data and KG data, and whether there is a many-to-many relationship between the source and target of the link.

We use the trained classifier to predict the probability of outgoing links of statement nodes in the candidate graph. The incoming link of a statement node will have the same score as the outgoing link representing the statement property. Finally, we use the Steiner Tree algorithm to select a subtree that maximizes the average likelihood of the links as our column relationship prediction. For example, assuming the scores of ⟨`Player`, `Team`, `P27`⟩, ⟨`Player`, `Team`, `P54`⟩, and ⟨`Player`, `Point`, `P1351`⟩ are 0.7, 0.85, and 0.6, respectively. Our Steiner Tree will select a subtree containing the last two links (`P54` and `P1351`) as it has the highest average likelihood.

4 Evaluation

4.1 Experiment Setup

Data: We evaluate our approach on three datasets for mapping tables to Wikidata: 250WT [37], HardTables (2022 SemTab challenge, round 1) [16], and WikidataTables (2023 SemTab challenge, round 1) [17]. The 250WT dataset contains 250 tables sampled from Wikipedia and is manually annotated. We update this dataset to remove all HTML markup (e.g., remove hyperlinks to the correct entities in Wikidata) because this information is generally not available for real-world tables. The HardTables and WikidataTables datasets are synthetic datasets containing 3650 and 9920 tables, respectively. We choose the datasets mapped to Wikidata ontology instead of datasets mapped to other knowledge graphs because Wikidata has a huge ontology with millions of classes and thousands of properties. Compared to other ontologies, the Wikidata ontology is designed to represent n-ary relationships (using statements and qualifiers), which can be found in real-world tables. Together, this captures the challenges and complexities of the semantic modeling task in the wild.

The most recent SemTab challenge (2023) introduces tFood, a new synthetic dataset generated from Wikidata's entities. However, we cannot use it because the columns' values containing entity names are anonymized using random characters. Since the names are replaced, this dataset focuses on a different aspect of the problem, which is finding the anonymous entities. This is not the focus of this paper and we leave it for future work. We also cannot use other datasets such as ToughTables [8] and WikiGS [12] (updated by [5]) because they do not have ground truth for column relationship annotation.

Our training dataset is created by randomly sampling five thousand automatically labeled tables from Section 3.1. We exclude any Wikipedia articles containing tables from the 250WT dataset from the training set to avoid data contamination. Also, we use Wikidata dumps on 2023-06-19 and Wikipedia dumps on 2023-06-20.

Evaluation Metrics: We assess the quality of the predictions in two different tasks: column type annotation (CTA) and column relationship annotation (CPA). The two tasks are evaluated using the approximate precision, recall, and F1 scores defined by the SemTab challenge [1]. The difference between the approximate metrics and their exact version is the partial credits they give when a system predicts a sub/parent class/property of the correct one. In particular, let $d(x)$ be the shortest distance of the predicted item (class or property) to the ground truth (GT) item. $d(x)$ is 0, 1 if the predicted item is equal to GT, or parent or child of GT, respectively. $\text{score}(x) = 0.8^{d(x)}$ 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 descendent of GT; otherwise, $\text{score}(x) = 0$. For readability, we sometimes refer to the approximate scores by just precision, recall, or F_1 score.

Baselines: We compare our approach, named GRAMS+, with winners of the SemTab challenges: MTab [29], KGCODE-TAB [24] and DAGOBAH [20], which

are the current state-of-the-art (SOTA) systems on this task. We create another baseline from GRAMS [37], the system that has the SOTA result on the 250WT dataset. Because GRAMS assumes that it is given the correct entities, we run GRAMS with the top 1 candidate entities. We cannot evaluate the best systems of the 2023 SemTab Challenge (TorchicTab [9] and SemTex [18]) as their source code is not available. However, SemTex reports its performance on the HardTables dataset, and we directly use the reported numbers for comparison.

To ensure a fair comparison, the baselines are modified to receive the same candidate entities as our method (100 candidate entities per cell). We cannot modify DAGOBAH because it is only available as an API at the time of submission. However, for DAGOBAH's CTA output, we notice that instead of predicting if a column contains people, they attempt to predict the occupation as the column type (e.g., politicians instead of humans). To give DAGOBAH credit for their prediction, if the column type in the ground truth is `Q5 human`, we will map the predicted occupation (subclass of `Q215627 person`) to the correct class.

Modeling Training: Our neural network models for entity linking and column relationship prediction are trained using Adam [22] with a learning rate of 1e−4 for 50 epochs.

4.2 Overall Performance

Table 1 shows the performance of our method, GRAMS+, versus the baselines on the three datasets: 250WT, HardTables, and WikidataTables. On the 250WT dataset, GRAMS+ outperforms MTab by 12.32% and 11.78% and DAGOBAH by 4.57% and 4.95% in macro-average approximate F_1 scores in the CPA and CTA tasks, respectively. Our method achieves considerably higher F_1 scores than GRAMS and KGCode-Tab in both tasks. In the CPA task, we generate candidate relationships using the method in [37]. This method does not rely on detecting subject columns of tables and can also detect n-ary relationships. Therefore, it helps us achieve higher recall. In addition, we also have greater precision because the neural network for scoring relationships is better at distinguishing incorrect relationships, thanks to the distant supervised training. The improvement in the CTA task is mainly due to better candidate entity scoring on ambiguous tables. For example, for the column `Team` in the motivating example, our entity linking model correctly ranks entities of national rugby teams among the top 2 (top 1 is national football teams). MTab, however, prefers entities of type country as it does not use any signal from the surrounding context (e.g., header). DAGOBAH also selects countries for this example.

On the HardTables and WikidataTables datasets (Table 1), most methods achieve high performance. The reason is that these synthetic datasets are much less noisy and less ambiguous than the 250WT dataset. The mentions and entity labels are often identical or differ by one or two characters, whereas in 250WT, the differences can be more significant (e.g., `Ireland` versus `Ireland National Rugby Union Team` or `CB` versus `center back`). DAGOBAH, however, has a much lower CPA recall than the other methods as their API turns off the

Table 1. Performance comparison between our method, GRAMS+, and the baselines on 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	KGCode-Tab	28.25	**70.01**	36.52	42.28	58.03	47.82
	GRAMS	75.86	42.27	44.71	66.33	77.38	70.45
	MTab	73.28	50.72	54.09	64.61	71.9	67.16
	DAGOBAH	76.42	60.92	61.84	71.47	**78.95**	73.99
	GRAMS+	**82.79**	62.06	**66.41**	**80.54**	78.26	**78.94**
Hard Tables	DAGOBAH	**99.2**	36.40	37.10	92.9	90.0	90.1
	KGCode-Tab	88.33	86.79	81.22	74.86	80.42	73.64
	GRAMS	95.64	83.72	84.57	80.37	87.64	80.54
	MTab	98.78	**94.06**	**94.1**	**94.87**	**91.5**	**91.53**
	GRAMS+	98.92	91.20	91.71	94.36	90.58	90.63
Wikidata Tables	MTab	**94.7**	**95.91**	**92.24**	**97.14**	93.51	**94.1**
	GRAMS+	94.38	91.10	89.25	96.08	**94.00**	94.06

feature to predict relationships to numeric columns. Also, we cannot evaluate DAGOBAH on the WikidataTables dataset as they discontinued their public API. While our method's results exceed DAGOBAH, they are lower than MTab by 2.38% and 0.9% in F_1 scores in CPA and CTA tasks on the HardTables dataset and by 2.99% and 0.9% in F_1 scores in CPA and CTA tasks on the WikidataTables dataset, respectively.

Analyzing the results of our approach, we find that most errors in the CPA task are due to predicting relationships of unannotated pairs of columns. For example, we need to predict the relationship `P361 part of` between columns 0 and 1, but GRAMS+ predicts the inverse relationship `P527 has part` between columns 1 and 0; hence, it does not get credit. To verify our finding, we ran another experiment in which GRAMS+ and MTab were given the target columns that the systems should predict. For example, an input table has three columns but the systems are instructed to predict only relationships between columns 0 and 1. In this experiment, on the HardTables dataset, both methods' new CPA average macro F_1 scores are 94.48%, and on the WikidataTables, our method has a better CPA F_1 score by 0.51%. Therefore, our neural network for ranking relationships is not the main reason for the lower performance than the baseline on this dataset.

For our CTA results, we see that many failed tables have only 3 or 4 rows. Either the entities' types in the rows are changed in our Wikidata snapshot (they no longer belong to the target classes in the ground truth), or we incorrectly rank some entities in the top 1. Thus, the score differences between the target classes and incorrect classes are very similar. Because our CTA method only relies on the candidate entity scores, it does not use our CPA task's output to help disambiguate them. This is a limitation of our CTA approach and we leave this for future work. Note that this issue tends not to happen for bigger tables.

Table 2. Performance comparison with the baselines when the target columns for CPA and CTA are provided. m-AP, m-AR, and m-AF_1 are micro-average approximate precision, recall, and F_1 in percentage, respectively

Dataset	Method	CPA			CTA		
		m-P	m-R	m-F_1	m-AP	m-AR	m-AF_1
Hard Tables	KGCode-Tab	98.19	86.85	92.17	86.73	81.40	83.98
	KGCode-Tab [24]	94.4	94.0	94.2	91.8	89.43	90.6
	SemTex [18]	-	-	97.05	-	-	93.85
	MTab	**99.01**	97.81	**98.4**	95.09	95.09	95.09
	DAGOBAH [20]	99	97.8	**98.4**	**97.5**	**97.5**	**97.5**
	GRAMS+	98.18	**97.93**	98.06	94.02	93.98	94.00
Wikidata Tables	SemTex [18]	-	-	96.4	-	-	93.4
	MTab	99.35	96.90	**98.11**	97.18	95.47	**96.32**
	GRAMS+	98.39	96.03	97.20	95.79	93.94	94.86

The reason is that our CTA method indirectly utilizes the collective signal from other columns since our entity linking method prefers entities with overlapping data with the table.

As we cannot obtain the source code of DAGOBAH and SemTex, we cannot ensure the same experiment setup (e.g., having the same candidate entities). Therefore, we evaluate our method and MTab in the same setting as the SemTab challenge to compare with their results. Results of this experiment are reported in Table 2. For DAGOBAH and SemTex, we directly use the numbers from their papers. On the HardTables, DAGOBAH outperforms the other systems in the CPA and CTA tasks by 2.52% and 3.6% (F_1 scores), respectively. One possible reason is the changes in the correct entities' types and properties in our Wikidata snapshot, as discussed above. The difference between the CPA performance of MTab and our method in this experiment is small and the difference is not statistically clear[4]. On the WikidataTables, MTab's performance surpasses all systems by at least 0.91% and 1.46% (F_1 scores) in CPA and CTA tasks, respectively. However, our method also outperforms SemTex, the system that has the highest performance among the SemTab2023 participants by 0.8% and 1.46% (F_1 scores) in CPA and CTA tasks, respectively.

4.3 Impact of Entity Linking on the Performance

To understand the impact of the entity linking step, we evaluate our method on two different entity ranking methods: (1) using our entity linking likelihood score and (2) using MTab's candidate retrieval score. In addition, we experiment with different numbers of candidate entities per cell.

Table 3 shows the performance of our system with the two aforementioned ranking approaches. The CTA F1 score drops significantly by 6.72%, while the CPA F1 score slightly decreases by 2%. The results indicate that the candidate

[4] The p-value of the sign test [11, 38] on the accuracies of the two systems is 0.086

Table 3. The performance of our method on the 250WT dataset with respect to two different candidate ranking methods

Method	CPA			CTA		
	AP	AR	AF_1	AP	AR	AF_1
Our Entity Linking Score	82.79	62.06	66.41	80.54	78.26	78.94
Retrieval Score	80.74	61.00	64.41	73.60	71.70	72.22

entity scores play a critical role in the CTA task. In the CPA task, our model can leverage other collective signals in the table to make more accurate predictions and, thus, is less dependent on the candidate scores.

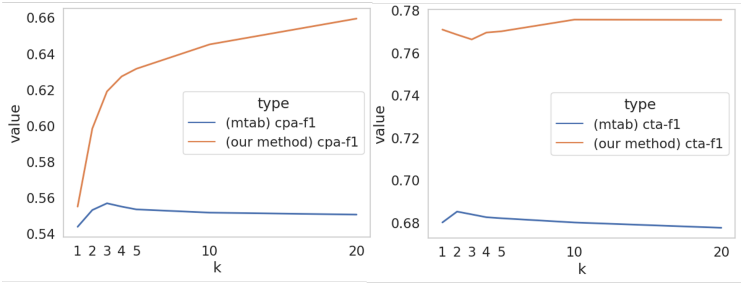


Fig. 4. Performances of our system and MTab with different numbers of candidate entities (x-axis) on the 250WT dataset. The CPA F_1 scores are shown in the left figure and the CTA F_1 scores are shown in the right

Figure 4 shows GRAMS+'s and MTab's performance with different numbers of candidate entities (K) per cell. In general, our CPA F_1 score increases as K increases, while MTab's CPA F_1 score peaks at $K = 3$ (55.7%) and then gradually declines. We also see a similar weaker trend in CTA scores. One of the reasons is that the input noise increases as the number of candidate entities increases. This demonstrates that our method is more robust to noise.

4.4 Impact of Table Metadata on the Performance

To understand how well our method can leverage the table metadata (column headers) to overcome ambiguous examples, we perform an ablation study by removing the table metadata from the entity-context similarity features described in Section 3.2. Table 4 shows that our approach can utilize the table metadata to overcome ambiguous cases; hence, our F_1 scores increase by 2.66% and 2.05% in CPA and CTA tasks, respectively. Without table metadata, our approach still outperforms the baselines by at least 1.91% and 2.9% in F_1 scores in CPA and CTA tasks, respectively.

5 Related Work

Previous work on the semantic modeling task includes supervised [35, 36, 13] approaches for custom domain ontologies and approaches that exploit KGs [25,

Table 4. The performance of our method on the 250WT dataset with and without table headers

Our Method	CPA			CTA		
	AP	AR	AF_1	AP	AR	AF_1
With metadata	82.79	62.06	66.41	80.54	78.26	78.94
Without metadata	77.64	61.29	63.75	77.67	77.18	76.89

28, 32, 39, 29, 20, 24, 9, 18]. The supervised approaches often take two inputs: a target ontology and a set of known semantic descriptions as training data. Taheriyan et al. [35] build a semantic description by finding a Steiner Tree that connects the data source's attributes, in which the Steiner Tree is a subgraph of the graph created by integrating known semantic descriptions in the training set. Building on this idea, Vu et al. [36] use a probabilistic graphical model (PGM) to compute the likelihood of a semantic description. Then, they use the PGM as a scoring function to search for the most probable semantic description. Recent work by Feng et al. [13] attempts to improve the prediction of [35] via post-processing using a domain KG. Despite being flexible in choosing a target ontology, these approaches suffer from the cold start problem: users need to label enough data sources before the systems can perform well. This issue is more profound with a large ontology as they would need lots of training data.

The approaches exploiting existing knowledge in KG are typically unsupervised. However, these approaches can also benefit from some labeled annotations to fine-tune their parameters, which are typically few. The common methodology of these approaches is to identify KG entities in a table (Entity Linking - EL) and match the properties of entities with values in the table to find column types (CTA) and relationships between columns (CPA). Limaye et al. [25] and Mulwad et al. [28] are among the first works to solve the three tasks (EL, CTA, and CPA) jointly using probabilistic graphical models. However, they do not handle non-entity columns in the tables. Later work shifts to the iterative paradigm and expands the problem setting to include literal columns. Ritze et al. [32] first identify a table's subject (entity) column, then find the candidate relationships between the subject column and other columns in the table. Ritze et al. iteratively update the candidate entities and candidate relationships until stability is achieved. Zhang et al. [39] refine entity linking results to be consistent with the annotated column types and the table's domain, estimated using a bag-of-words method, and then predict column relationships. The best performance systems of the SemTab challenges [21, 7, 1, 15], such as MTab [29], DAGOBAH [20], and others KGCode-Tab [24], LinkingPark [6], BBW [33], TorchicTab-Heuristic [9], and SemTex [18] also follow the iterative paradigm. Their systems improve various aspects of the pipeline, such as candidate entity retrieval and scoring functions to rank the matched results. GRAMS [37] extends to the full setting to predicting n-ary relationships and missing context values for Wikipedia tables. In particular, they build a candidate graph of potential relationships and use a Probabilistic Soft Logic [2] to rank the graph edges and select the most probable subgraph containing relationships in the input table. Our work tackles the

complete setting of the semantic modeling task and differs from previous work in using distant supervision to automatically generate labeled data to train machine learning models instead of relying on hand-crafted scoring functions. It is the first approach to use distant supervision for CPA.

As entity linking plays a vital role in many semantic modeling approaches, it is worth mentioning that using distant supervision for table entity linking has been explored in previous work. Bhagavatula et al. [3] use hyperlinks in Wikipedia tables as the ground truth to train a logistic regression model for collectively disambiguating candidate entities. DAGOBAH [20] also use the dataset from [3] to learn a relevance score between entity descriptions and column headers using a transformer model. Compared to the previous methods, our approach excludes hyperlinks inconsistent with the context instead of using all of them. This helps reduce noise and makes learning easier. Using a transformer model to learn a relevance score with the headers in DAGOBAH is similar to the context feature in our entity linking model. However, we directly train the model to disambiguate the candidate entities rather than just learning a relevance score.

Finally, there is some previous work on table understanding that has different or restricted settings compared to the semantic modeling task in this paper [26]. For example, DSL [30], ColNet [4], Sherlock [19] focus on only annotating the column types in the tables. Luzuriaga et al. [27] focus on generating triples of relationships between columns for Wikipedia tables. TURL [10], DODUO [34], and TorchicTab-Classification [9] predict both column types and column relationships. However, the three approaches (TURL, DODUO, and TorchicTab-Classification) need to be provided with the target columns to predict, while in our setting this information is not given and the method figures it out. They treat this problem as a supervised classification problem, and all use transformer-based models. TorchicTab-Classification extends DODUO with a sub-table sampling strategy to work for large tables and is trained on SOTAB [23], a dataset constructed automatically from microdata in websites. TURL and DODUO are trained on a modified version of WikiGS [12], in which Wikipedia tables are labeled automatically with Freebase ontology. The labels for WikiGS are generated based on the common types and relationships of entities found in the tables. While the updated WikiGS can be used to train our methods, we developed a new automatically labeled dataset from Wikipedia to help teach models to handle challenges in the real world, such as n-ary relationships (e.g., number of goals a player scored for different sports teams) or ambiguous tables (e.g., cells should be linked to national sports teams but are incorrectly linked to countries in Wikipedia). Our primary contribution is not the auto-labeled dataset, but rather it is a novel semantic modeling method that can map tables to a target knowledge graph. Although TURL and DODUO are trained on an auto-labeled dataset similar to ours, their approaches require the same sets of predefined target classes and properties as target labels for both training and testing datasets. Our approach does not require testing datasets to have the same target labels. For instance, approximately 59% (89/150) and 26% (31/119) of the classes and properties, respectively, in the 250WT dataset do not appear in our training set.

6 Conclusion and Future Work

This paper presents a novel distant supervision approach for learning semantic descriptions of tables. Using the hyperlinks in Wikipedia tables, we generate labeled examples to train two neural networks to predict the likelihood of candidate entities and column relationships. Then, we use the two models to predict column types and relationships to obtain the semantic descriptions of tables. The empirical evaluation shows that our approach outperforms state-of-the-art methods on a large set of real-world tables. Moreover, it is more robust to noise from the entity linking step and can leverage the table metadata to overcome ambiguous examples.

We demonstrate that we can use automatically labeled tables to train a semantic modeling method to achieve good results without hand-crafted scoring functions. To achieve these results, or method does make some assumptions. First, our method assumes that some of the entities in a table also appear in a target KG (data overlapping assumption). Thus, our method cannot be used to map tables to a domain ontology (e.g., CIDOC-CRM[5]) that does not have a corresponding knowledge graph with overlapping data. Second, we assume we can find relevant contextual information in order to map the data in a table to the correct ontology class. Although our method has incorporated table metadata (column headers) to our predictions, information from column headers may not be distinct enough in some cases (e.g., the header `team` can mean both national soccer teams or national rugby teams). A more sophisticated approach, such as one using large language models, can be used to better understand the surrounding context. We leave this for future work.

In future work, we plan to increase the number of labeled tables by relaxing filtering conditions described in Section 3.1 and creating new tables by adding or removing rows in the labeled Wikipedia tables. Additionally, learning to jointly predict column relationships and types can potentially improve the performance of the whole task. Another exciting direction using the auto-labeled dataset is to explore methods that can leverage the table context and metadata to model the tables that have little or no overlap with a KG.

Supplemental Material Statement: The experiment results, code, and models are available at https://purl.org/gramsplus.

Acknowledgements

This material is based upon research 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 the Defense Advanced Research Projects Agency (DARPA); or its Contracting Agent, the U.S. Department of the Interior, Interior Business Center, Acquisition Services Directorate, Division V.

[5] https://www.cidoc-crm.org/

References

1. Abdelmageed, N., Chen, J., Cutrona, V., Efthymiou, V., Hassanzadeh, O., Hulsebos, M., Jimenez-Ruiz, E., Sequeda, J., Srinivas, K.: Results of semtab 2022. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2022). CEUR Workshop Proceedings, vol. 3320. CEUR (January 2022), https://openaccess.city.ac.uk/id/eprint/29744/
2. Bach, S.H., Broecheler, M., Huang, B., Getoor, L.: Hinge-loss markov random fields and probabilistic soft logic. Journal of Machine Learning Research **18**(109), 1–67 (2017)
3. Bhagavatula, C.S., Noraset, T., Downey, D.: TabEL: Entity linking in web tables. In: The Semantic Web - ISWC 2015. pp. 425–441. Springer International Publishing (2015)
4. 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 (Jul 2019)
5. Chen, J., Jimenez-Ruiz, E., Horrocks, I., Sutton, C.: Learning semantic annotations for tabular data. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence Organization, California (Aug 2019). https://doi.org/10.24963/ijcai.2019/289
6. Chen, S., Karaoglu, A., Negreanu, C., Ma, T., Yao, J.G., Williams, J., Gordon, A., Lin, C.Y.: LinkingPark: An integrated approach for semantic table interpretation. http://ceur-ws.org/Vol-2775/paper7.pdf, accessed: 2023-10-6
7. Cutrona, V., Chen, J., Efthymiou, V., Hassanzadeh, O., Jimenez-Ruiz, E., Sequeda, J., Srinivas, K., Abdelmageed, N., Hulsebos, M., Oliveira, D., Pesquita, C.: Results of semtab 2021. In: 20th International Semantic Web Conference. vol. 3103, pp. 1–12. CEUR Workshop Proceedings (March 2022), https://openaccess.city.ac.uk/id/eprint/28056/
8. Cutrona, V., Bianchi, F., Jiménez-Ruiz, E., Palmonari, M.: Tough Tables: Carefully Evaluating Entity Linking for Tabular Data (Jan 2023). https://doi.org/10.5281/zenodo.7419275
9. Dasoulas, I., Yang, D., Duan, X., Dimou, A.: TorchicTab: Semantic Table Annotation with Wikidata and Language Models. In: CEUR Workshop Proceedings. pp. 21–37 (2023)
10. Deng, X., Sun, H., Lees, A., Wu, Y., Yu, C.: TURL: Table understanding through representation learning (Jun 2020)
11. Dixon, W.J., Mood, A.M.: The statistical sign test. Journal of the American Statistical Association **41**(236), 557–566 (1946)
12. 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
13. Feng, Z.W., Xu, J.K., Mayer, W., Huang, W.Y., He, K.Q., Stumptner, M., Grossmann, G., Zhang, H.Y., Ling, L.: Automatic semantic modeling for structural data source with the prior knowledge from knowledge graph (2021)
14. Garbe, W.: Symspell (6 2012), https://seekstorm.com/blog/1000x-spelling-correction/
15. Hassanzadeh, O., Abdelmageed, N., Efthymiou, V., Chen, J., Cutrona, V., Hulsebos, M., Jiménez-Ruiz, E., Khatiwada, A., Korini, K., Kruit, B., et al.: Results of semtab 2023. In: CEUR Workshop Proceedings. vol. 3557, pp. 1–14 (2023)

16. Hassanzadeh, O., Efthymiou, V., Chen, J.: SemTab 2022: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching Data Sets - "Hard Tables" R1 and R2 (Dec 2022). https://doi.org/10.5281/zenodo.7416036
17. Hassanzadeh, O., Efthymiou, V., Chen, J.: SemTab 2023: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching Data Sets - "Wikidata Tables" R1 (Sep 2023). https://doi.org/10.5281/zenodo.8393535
18. Henriksen, E.G., Khorsid, A.M., Nielsen, E., Stück, A.M., Sørensen, A.S., Pelgrin, O.: Semtex: A hybrid approach for semantic table interpretation (2023)
19. Hulsebos, M., Hu, K., Bakker, M., Zgraggen, E., Satyanarayan, A., Kraska, T., Demiralp, Ç., Hidalgo, C.: 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, NY, USA (Jul 2019)
20. 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)
21. Jimenez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., Srinivas, K., Cutrona, V.: Results of semtab 2020. CEUR Workshop Proceedings **2775**, 1–8 (January 2020), http://ceur-ws.org/Vol-2775/
22. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (Dec 2014)
23. 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)
24. Li, X., Wang, S., Zhou, W., Zhang, G., Jiang, C., Hong, T., Wang, P.: KGCODE-Tab results for SemTab 2022. https://ceur-ws.org/Vol-3320/paper5.pdf, accessed: 2023-10-6
25. 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 (Sep 2010)
26. 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. Journal of Web Semantics **76**, 100761 (Apr 2023)
27. Luzuriaga, J., Munoz, E., Rosales-Mendez, H., Hogan, A.: Merging web tables for relation extraction with knowledge graphs. IEEE Trans. Knowl. Data Eng. pp. 1–1 (2021)
28. Mulwad, V., Finin, T., Joshi, A.: Semantic message passing for generating linked data from tables. In: The Semantic Web – ISWC 2013. pp. 363–378. Springer Berlin Heidelberg (2013)
29. 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: 2023-10-6
30. Pham, M., Alse, S., Knoblock, C.A., Szekely, P.: Semantic labeling: A Domain-Independent approach. In: The Semantic Web – ISWC 2016. pp. 446–462. Springer International Publishing (2016)
31. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using siamese BERT-Networks (Aug 2019)
32. 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, NY, USA (Jul 2015)

33. 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: 2023-10-6
34. Suhara, Y., Li, J., Li, Y., Zhang, D., Demiralp, Ç., Chen, C., Tan, W.C.: Annotating columns with pre-trained language models. In: Proceedings of the 2022 International Conference on Management of Data. ACM, New York, NY, USA (Jun 2022)
35. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: Learning the semantics of structured data sources. Journal of Web Semantics **37-38**, 152–169 (Mar 2016)
36. 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, NY, USA (May 2019)
37. Vu, B., Knoblock, C.A., Szekely, P., Pham, M., Pujara, J.: A Graph-Based approach for inferring semantic descriptions of wikipedia tables. In: The Semantic Web – ISWC 2021. pp. 304–320. Springer International Publishing (2021)
38. Yeh, A.: More accurate tests for the statistical significance of result differences. In: COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics (2000), https://aclanthology.org/C00-2137
39. Zhang, Z.: Effective and efficient semantic table interpretation using TableMiner+. Semant. Web **8**(6), 921–957 (Aug 2017)