A Graph-Based Approach for Inferring Semantic Descriptions of Wikipedia Tables

Binh Vu(✉), Craig A. Knoblock, Pedro Szekely, Minh Pham, and Jay Pujara

USC Information Sciences Institute, Marina Del Rey, CA 90292, USA
{binhvu,knoblock,pszekely,minhpham,jpujara}@isi.edu

Abstract. There are millions of high-quality tables available in Wikipedia. These tables cover many domains and contain useful information. To make use of these tables for data discovery or data integration, we need precise descriptions of the concepts and relationships in the data, known as semantic descriptions. However, creating semantic descriptions is a complex process requiring considerable manual effort and can be error prone. In this paper, we present a novel probabilistic approach for automatically building semantic descriptions of Wikipedia tables. Our approach leverages hyperlinks in a Wikipedia table and existing knowledge in Wikidata to construct a graph of possible relationships in the table and its context, and then it uses collective inference to distinguish genuine and spurious relationships to form the final semantic description. In contrast to existing methods, our solution can handle tables that require complex semantic descriptions of n-ary relations (e.g., the population of a country in a particular year) or implicit contextual values to describe the data accurately. In our empirical evaluation, our approach outperforms state-of-the-art systems on the SemTab2020 dataset and outperforms those systems by as much as 28% in F1 score on a large set of Wikipedia tables.

Keywords: Semantic models · Semantic descriptions · Knowledge graphs · Probabilistic soft logic · Semantic web · Linked data · Ontology

1 Introduction

Wikipedia is one of the largest encyclopedias in the world. Extracting and integrating the structured data from Wikipedia to knowledge graphs (KGs) can bring great benefits to many applications. DBpedia, a popular KG, has shown the success and impact of such a strategy, but only uses infoboxes. Besides these infoboxes, Wikipedia also has millions of high-quality tables covering a wide range of domains. Leveraging these tables can potentially help to add or keep the knowledge in KGs up-to-date. For example, in an evaluation dataset collected from Wikipedia (Sect. 4.1), we found that approximately 64% of the relationships in the data in these tables are not present in Wikidata. However, it is challenging to make use of these tables on a large scale as they are stored in different schemas. The task of building semantic descriptions of tables (also

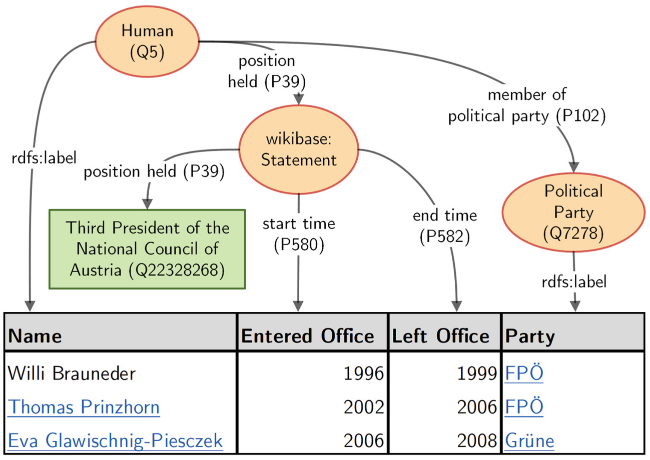
A. Hotho et al. (Eds.): ISWC 2021, LNCS 12922, pp. 304–320, 2021.
https://doi.org/10.1007/978-3-030-88361-4_18

called semantic modeling [19,20]) addresses this challenge by precisely describing concepts and relationships contained in the data in a machine-readable form. A semantic description of a table is a graph where each node represents either an ontology class, a column, an entity, or a literal (e.g., number, text, or date), and each edge represents an ontology property encoding a relationship between the two nodes (Fig. 1). From the semantic description, we can automatically generate mapping rules of mapping languages such as RML [4] or D-REPR [21] to convert the table data to RDF triples to import into KGs.

Since creating semantic descriptions requires significant effort and expertise [9], there are many studies to address this problem. Generally, they can be placed into two groups. The first group is supervised methods trained on a set of known semantic descriptions with given domain ontologies [19,20]. These methods are difficult to apply to the Wikipedia tables as there is little training data available. The second group is methods that utilize KGs such as DBpedia [3,16] or Wikidata [13,18] to integrate data from tables. The intuition of these approaches is that the overlap between entities in a table with entities in KGs can be used to recover the semantic description of the table. Specifically, by matching the property values of the overlapped entities with other cells in the table, they can predict binary relationships between two columns based on the matched properties and column types using the types of the entitites.

Approaches in the second group can be applied to map the Wikipedia tables and generally do not require retraining their systems when the KG ontology is updated. However, they have two main limitations. First, their methods only consider values inside the tables but not values in the surrounding context. We found in many tables that the implicit contextual values are critical to understanding the semantics of a table. For example, a table about cast members of a movie and their roles typically does not have the movie in the table data as it is mentioned in the context. Second, they do not deal with n-ary relations needed to accurately and fully represent knowledge in the tables. Examples of n-ary relations are a politician elected to an office position from an electoral district or sales of a company reported in a particular year.

To address these issues, we present a new approach for semantic modeling that uses graphs to represent possible (n-ary) relationships in the tables and collective inference to eliminate spurious relationships on the graphs. Specifically, we construct a candidate graph containing relationships between table columns and its context values by leveraging possible connections between data in the table and existing knowledge in Wikidata. Then, incorrect relationships in the candidate graph are detected and removed using a Probabilistic Soft Logic (PSL) [1] model. Through collective inference, the PSL model favors links with high confidence, more informative, and consistent with constraints in the ontology and existing knowledge in Wikidata. To assess the effectiveness of our method, we evaluate our method on a real-world dataset for mapping tables to Wikidata and on the dataset from the SemTab2020 challenge [6]. These experiments show that our method outperforms the state-of-the-art systems on all datasets, with an improvement of 28% F_1 score on the real-world dataset.



Name	Entered Office	Left Office	Party
Willi Brauneder	1996	1999	FPÖ
Thomas Prinzhorn	2002	2006	FPÖ
Eva Glawischnig-Piesczek	2006	2008	Grüne

Fig. 1. A table of third presidents of the National Council of Austria with its semantic description on top. The node `wikibase:Statement` is used to represent an n-ary relationship of a position (Third President) and its start and end time. The position is not in the table but is introduced via an entity (the green node). (Color figure online)

The contribution of this paper is a novel graph-based method for semantic modeling that collectively determines correct relationships between two or more columns and implicit contextual values using PSL. Our solution offers two key technologies: (i) an algorithm to construct a graph of plausible semantic descriptions of tables using external knowledge from Wikidata and (ii) a probabilistic model that utilizes features from external knowledge and related relationships in the graph for robust relationship prediction.

2 Motivating Example

In this section, we explain the problem by giving an example of mapping a real table about the third presidents of the National Council of Austria to Wikidata. This example is also used throughout the paper to illustrate the steps of our approach. We also interchangeably refer to Wikidata entities, classes (Qnodes), and properties (Pnodes) either by their labels and ids (e.g., `Human (Q5)`) or just by their ids (e.g., `Q5`).

Figure 1 shows a snippet of the table at the bottom and its semantic description on the top. In the figure, yellow nodes represent ontology classes, green nodes represent entities or literals, and edges are ontology properties. For example, the link `rdfs:label` between the node `Human (Q5)` and the first column depicts that each cell in the column is a person whose name is specified in the value of that cell. Similarly, the link `P102` between the class `Human (Q5)` and `Political Party (Q7278)` states that each person is a member of the corresponding political party. The property `position held (P39)` connects the node `Q5` to an entity `Q22328268` and columns `Entered Office` and `Left Office` to describe the time each person holds the third president position. This is an n-ary relationship and is represented by an intermediate `wikibase:Statement` node. Note that in Wikidata every claim is represented as a statement, so there is a statement node for the relationship `P102` of node `Q5` and node `Q7278`. However, since this is a binary relationship, we have omitted the statement node for conciseness.

In the table, some cells are linked to Wikipedia articles such as `Eva Glawischnig-Piesczek` (3rd row, 1st column). By querying Wikidata to obtain a Qnode associated with the `Eva Glawischnig-Piesczek` article, we know that she was the `third president of the National Council of Austria (Q22328268)` from `2006` to `2008`. As the information appears in the same row of the 2nd and 3rd columns, this suggests that `start time (P580)` and `end time (P582)` could be the relationships between those columns and of an n-ary relationship `position held (P39)` of `Q22328268`. Following this process, we may discover in the second row that `Thomas Prinzhorn` was a second president, and he left the office in `2002`, while there may be no suggestions from data in the 1st row as `Wilhelm Brauneder` does not link to any Qnode.

From this example, we observe that matching table data to KGs can suggest correct semantic descriptions. Yet, predictions solely relying on data matching can be imprecise. To go beyond simple data matching, we develop a graph-based approach that uses a probabilistic graphical model to combine evidence from external knowledge and related possible matched relationships to predict the most probable semantic description.

3 Building Semantic Descriptions of Tables

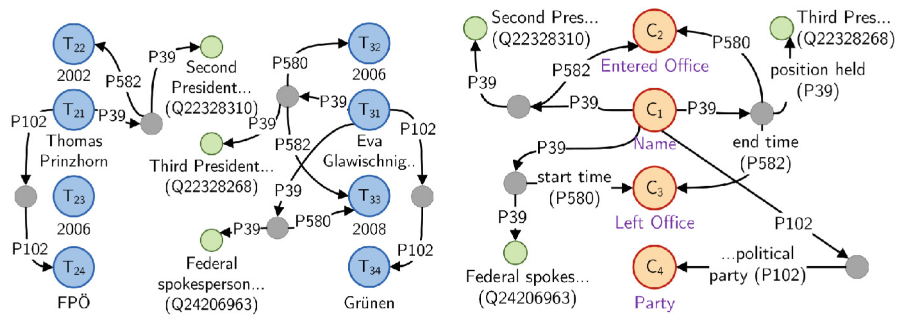
The problem of finding semantic descriptions of Wikipedia tables is defined as follows. Let T be a linked relational table, in which a cell $c_{i,j}$ of row i, column j may link to entities in a target KG, $\mathcal{C} = \{v_1, v_2, ..., v_n\}$ be a set of values (literals or entities in KG) found in the surrounding context of T. We want to find the semantic description $sm(T, \mathcal{C})$ of T with respect to its context $\mathcal{C}$.

Our approach consists of two main steps. The first step is to build a candidate graph of relationships between columns and context values. Then, we use collective inference to identify correct relationships and correct types of columns containing entities (called entity columns) to create a final semantic description.

Preprocessing. Since we use Wikidata as the target KG, the semantic description will be described in terms of the Wikidata ontology. Classes of the ontology are all Qnodes participating in the `subclass of (P279)` relationship, and properties of the ontology are all Wikidata properties and the `rdfs:label` property. Also, cells in the Wikipedia tables are not directly linked to Wikidata entities but have hyperlinks to Wikipedia articles. Thus, we apply a preprocessing step to automatically convert the hyperlinks to Wikipedia articles to Wikidata entities using Wikidata sitelinks.

3.1 Constructing Candidate Graphs

To create a candidate graph, we first create a data graph of all possible relationships between table cells and table context. Then, we summarize the data



(a) Data graph (b) Candidate graph

Fig. 2. Excerpts of a data graph and a candidate graph built from the data graph for the table in Fig. 1. Some edges are displayed without their full labels for readability. Grey, blue, yellow, and green nodes are statements, table cells, table columns, and entities, respectively.

graph to obtain relationships between columns and table context. This approach makes it easy to handle n-ary relations and infer missing links.

Constructing Data Graphs. Algorithm 1 outlines the process of building a data graph. To begin with, we add cells of T and items in $\mathcal{C}$ as nodes to an empty graph G_d (line 1–5). Then, we find paths in Wikidata that connect two nodes in G_d using two functions `FindEnt2EntPaths` and `FindEnt2LiteralPaths` (lines 8–11). The former function simply returns paths between two entities in Wikidata. The latter function returns paths from an entity to a literal. Since literals in the table are not always matched exactly with the corresponding values in the KG, we "fuzzy" match literals depending on their types. For example, numbers are matched if they are within a 5% range; dates are matched if they are equal or their years are equal (when the literals only have years); strings are matched if they are the same. The function `FindEnt2EntPaths` has an extra parameter *max_hop* controlling the length of discovered paths. If the maximum hop is two, a path can reach a target literal or entity via an intermediate entity in Wikidata. If the target entity or literal is found in qualifiers of statements, we also return extra paths from the source entity to the statements' values in order to comply with the Wikidata data model. Note that we only need to find paths between pairs of nodes that can be linked (line 7). Two nodes are linkable when they are cells of the same record (i.e., in the same row), or one node is a cell and the other node is a value in the context.

The discovered paths will then be added to G_d such that the original identifiers of Wikidata statements and entities are preserved (line 12). This allows paths of n-ary relationships to be connected automatically as they share the same Wikidata statements. Figure 2a shows an excerpt of the data graph for the table in the motivating example after this step.

Algorithm 1: CONSTRUCT DATA GRAPH

```
   Input: Input table $T$, Input context $\mathcal{C} = \{v_1, ..., v_n\}$, Max hop $max\_hop$
   Output: the data graph $G_d$
1  $G_d \leftarrow$ empty graph
2  add cells in the table ($c_{i,j} \in T$.cells) as nodes to $G_d$
3  add values in the context ($v_i \in \mathcal{C}$) as nodes to $G_d$
4  newPaths $\leftarrow$ []
5  for $n_i \leftarrow G_d$.nodes do
6      for $n_j \leftarrow G_d$.nodes do
7          if CanLink ($n_i$, $n_j$) then
8              for $e_i \leftarrow n_i$.linkedEntities do
9                  for $e_j \leftarrow n_j$.linkedEntities do
10                     add FindEnt2EntPaths ($e_i$, $e_j$, $max\_hop$) to newPaths
11                 add FindEnt2LiteralPaths ($e_i$, $n_j$) to newPaths
12 AddPaths ($G_d$, newPaths)
13 InferMissingLinks ($G_d$)
14 return $G_d$
```

Finally, we run inference on G_d to complete missing links based on logical rules specified in the Wikidata ontology. Specifically, our ad-hoc rule-based reasoner uses sub-property, inverse, and transitive rules. The intuition is that the final graph G_d after inference should be the same as if we run inference on the KG, then build the data graph G_d.

Constructing Candidate Graphs. With the data graph G_d built from the previous step, we will summarize it to create a super graph of plausible semantic descriptions. The step is similar to a reversion of the process that generates an RDF graph from the semantic description of the table. Specifically, relationships of cells of two columns or of cells of a column and a context value are consolidated if they are of the same property. For example, in Fig. 2b, relationships `P102` between cells of columns 1 and 2 are grouped to be represented as one edge `P102` between these columns in the graph.

This idea is implemented in Algorithm 2. It starts by adding columns in the tables (line 1–2) as nodes to G_s. Then, we add literal or entities nodes in G_d to G_s keeping their original id (line 3). Next, for each pair of nodes (u_d and v_d) in G_d in which v_d is the value of a property e of u_d specified by statement node $\texttt{stmt}_d$ (lines 4–7), we find the corresponding nodes of u_d and v_d in G_s called u_s and v_s, respectively (line 8). If u_d is a cell node, then its corresponding node u_s in G_s will be the column node; otherwise, u_s will be the node of the same id. Next, we add a new statement node $\texttt{stmt}_s$ of the relationship e between u_s and v_s if it does not exist (lines 9–11). After that, we add new qualifiers to $\texttt{stmt}_s$ based on qualifiers of $\texttt{stmt}_d$ with a similar manner (line 13–16).

Algorithm 2: CONSTRUCT CANDIDATE GRAPH

```
Input: A data graph $G_d$, Input table $T$
Output: the candidate graph $G_s$
1  $G_s \leftarrow$ empty graph
2  add columns in the table $T$ as nodes to $G_s$
3  add literal or entity nodes in $G_d$ as nodes to $G_s$, keeping their original id
4  for $u_d \leftarrow G_d$.nodes do
5      for $v_d \leftarrow G_d$.nodes do
6          if $v_d$ is value of a property $e$ of $u_d$ then
7              stmt$_d \leftarrow$ statement of property $e$ linking $u_d$ and $v_d$
8              $u_s$, $v_s \leftarrow$ corresponding node of $u_d$, $v_d$ in $G_s$, respectively
9              stmt$_s \leftarrow$ statement of property $e$ linking $u_s$ and $v_s$
10             if stmt$_s$ does not exist then
11                 add stmt$_s$ to $G_s$ and link $u_s$ to $v_s$: $u_s \xrightarrow{e}$ stmt$_s \xrightarrow{e} v_s$
12             for qualifier $q$ of stmt$_d$ do
13                 $t_d \leftarrow$ target node of $q$ of stmt$_d$ in $G_d$
14                 $t_s \leftarrow$ corresponding node of $t_d$ in $G_s$
15                 if the qualifier stmt$_s \xrightarrow{q} t_s$ does not exist then
16                     add qualifier stmt$_s \xrightarrow{q} t_s$ to $G_s$
17 return $G_s$
```

3.2 Predicting Correct Relationships Using PSL

The candidate graph obtained from the previous step can contain spurious relationships. To identify correct relationships, we use PSL [1]. PSL is a machine learning framework for developing probabilistic graphical models using first-order logic. A PSL model consists of predicates and rules (logic or arithmetic) constructed from those predicates. An example of a PSL rule is as follow:

$$w : \textsc{CloseFriend}(A, B) \wedge \textsc{CloseFriend}(B, C) \Rightarrow \textsc{Friend}(A, C)$$

where w is weight of the rule, CLOSEFRIEND, FRIEND are predicates, A, B, C are variables. The example rule can be read as "if A and B are close friends and B and C are close friends, then A and C should be friends". If a rule in PSL does not have weight, it will be considered as a hard constraint. Given a set of predicates' values called observations, PSL substitutes (or grounds) predicates in the rules with the observations and performs convex optimization to infer values of the unobserved predicates.

PSL Model. Table 1 shows the list of main predicates in our PSL model. $\textsc{CorrectRel}(N_1, N_2, P)$ and $\textsc{CorrectType}(N, T)$ are the target predicates that we want PSL to infer the values. With these predicates, we design the following PSL rules.

$$\neg\textsc{CorrectRel}(N_1, N_2, P_1) \tag{1}$$

$$\neg\textsc{CorrectType}(N, T) \tag{2}$$

$$\textsc{CanRel}(N_1, N_2, P) \wedge \textsc{PosRelFeat}_i(N_1, N_2, P) \Rightarrow \textsc{CorrectRel}(N_1, N_2, P) \tag{3}$$

$$\textsc{CanRel}(N_1, N_2, P) \wedge \textsc{NegRelFeat}_i(N_1, N_2, P) \Rightarrow \neg\textsc{CorrectRel}(N_1, N_2, P) \tag{4}$$

$$\textsc{CanType}(N, T) \wedge \textsc{PosTypeFeat}_i(N, T) \Rightarrow \textsc{CorrectType}(N, T) \tag{5}$$

$$\begin{aligned}&\textsc{CanRel}(N_0, S, P) \wedge \textsc{Statement}(S) \wedge \textsc{CanRel}(S, N_1, P) \wedge \textsc{CanRel}(S, N_2, Q)\\ &\quad \wedge N_1 \neq N_2 \wedge \neg\textsc{CorrectRel}(S, N_1, P) \Rightarrow \neg\textsc{CorrectRel}(S, N_2, Q)\end{aligned} \tag{6}$$

$$\begin{aligned}&\textsc{CanRel}(N_1, S_1, P_1) \wedge \textsc{Statement}(S_1) \wedge \textsc{CanRel}(S_1, N_2, P_1)\\ &\quad \wedge \textsc{CanRel}(N_1, S_2, P_2) \wedge \textsc{Statement}(S_2) \wedge \textsc{CanRel}(S_2, N_2, P_2)\\ &\quad \wedge \text{SubProp}(P_1, P_2) \Rightarrow \neg\textsc{CorrectRel}(N_1, S_2, P_2)\end{aligned} \tag{7}$$

$$\begin{aligned}&\textsc{CanRel}(N_1, S_1, P_1) \wedge \textsc{Statement}(S_1) \wedge \textsc{CanRel}(S_1, N_2, P_1)\\ &\quad \wedge \textsc{CanRel}(N_1, S_2, P_2) \wedge \textsc{Statement}(S_2) \wedge \textsc{CanRel}(S_2, N_2, P_2)\\ &\quad \wedge \text{SubProp}(P_1, P_2) \Rightarrow \neg\textsc{CorrectRel}(S_2, N_2, P_2)\end{aligned} \tag{8}$$

$$\begin{aligned}&\textsc{CanRel}(N_1, N_2, P_1) \wedge \textsc{CanRel}(N_2, N_3, P_2) \wedge \textsc{CorrectRel}(N_2, N_3, P_2)\\ &\quad \wedge \textsc{OneToMany}(N_2, N_3) \Rightarrow \neg\textsc{CorrectRel}(N_1, N_2, P_2)\end{aligned} \tag{9}$$

Rules 1 and 2 are default negative priors indicating that usually there is no relationship between two nodes and no type of column, respectively. Rules 3 and 4 state that if there is a link (N_1, N_2, P) between two nodes in G_s and there is a feature supporting or opposing the link, then the relationship (N_1, N_2, P) should be correct or incorrect, respectively. The supporting and opposing features of (N_1, N_2, P) are computed based on the number of rows in which we discover the relationship (N_1, N_2, P) (denoted as $\mathit{match}(N_1, N_2, P)$), and the number of rows in which existing data of the relationship in Wikidata is different from the data in the table (denoted as $\mathit{difference}(N_1, N_2, P)$). The two numbers are normalized in various ways: divided by the number of rows, number of rows that have entities, or by $\sum_p \mathit{match}(N_1, N_2, p) + \mathit{difference}(N_1, N_2, p)$ resulting in different features. Similar to rule 3, rule 5 also uses features to predict if T is a correct type of column N. Currently, it uses one feature which is the percentage of rows containing entities of type T.

Table 1. Predicates in the PSL model

Predicates	Meaning
$\textsc{CanRel}(N_1, N_2, P)$	A candidate relationship P between nodes N_1 and N_2
$\textsc{CorrectRel}(N_1, N_2, P)$	Denoting if a relationship (N_1, N_2, P) is correct
$\textsc{CanType}(N, T)$	A candidate type T of column N
$\textsc{CorrectType}(N, T)$	Denoting if the type column N is T
$\textsc{SubProp}(P_1, P_2)$	Property P_1 is a subproperty of P_2
$\textsc{Statement}(N)$	Node N in the candidate graph is a statement node
$\textsc{PosRelFeat}_i(N_1, N_2, P)$	Value of feature i backing the relationship (N_1, N_2, P)
$\textsc{NegRelFeat}_i(N_1, N_2, P)$	Value of feature i opposing the relationship (N_1, N_2, P)
$\textsc{PosTypeFeat}_i(N, T)$	Value of feature i backing the column type (N, T)
$\textsc{OneToMany}(N_1, N_2)$	A value in column N_1 is associated with multiple values in column N_2

Different from the previous rules, rules 6, 7, 8, 9 are applied to a group of relationships. They are used to enforce consistency of the descriptions with the Wikidata data model as well as to introduce inductive bias or prior knowledge of the desired semantic descriptions. Specifically, rule 6 states that if a property of a statement is inferred to be false, then the statement's qualifiers should also be false. Rules 7 and 8 favor fine-grain properties. Finally, rule 9 prefers that properties' values of non-subject entities should have a one-to-one correspondence to the entities. The non-subject entities are defined as entities with incoming relationships from other entities in the table (i.e., not the main entities that the table is about).

We use the same weight ($w = 2$) for all rules, except that the default negative priors (rules 1 and 2) should have less weight as instructed in PSL tutorial ($w = 1$); rules that introduce preferences should have very small weights (rules 7 and 8 have $w = 0.1$); and rules that act as constraints should have very high weights (rule 9 has $w = 100$).

Inference and Post-processing. From G_s, we extract values of all predicates in the PSL model except the CorrectRel and CorrectType predicates. Then, we run PSL inference to determine the values of the two predicates which represent the probabilities of links between nodes in G_s and types of columns, respectively. Values that have probabilities lower than a chosen threshold (0.5) are considered incorrect and are removed.

After running inference, there could be more than one correct link between two nodes. For instance, the PSL model predicts that `capital (P36)`, `capital of (P1376)`, or `located in...(P131)` are correct relationships for `Capital City` and `Country`. Thus, we run a post-processing step that selects only one path between two nodes such that it maximizes the sum of probabilities of relationships in the final semantic description while maintaining the tree structure of the description.

4 Evaluation

4.1 Datasets for Semantic Modeling

Our objective is to assess the ability of our method to infer correct semantic descriptions of linked tables. There are several standard datasets for benchmarking this problem, such as T2D [17] or Limaye [10]. However, these datasets are not linked to Wikidata; they are relatively simple and do not capture the complexity of the semantic modeling problem in Wikipedia tables. Therefore, we introduce a new dataset of 250 Wikipedia tables, called 250 WT, with their semantic descriptions built using the Wikidata ontology.

The new dataset's tables are selected from a pool of 2 million relational Wikipedia tables with the following procedure to ensure good coverage over multiple domains and produce high-quality unambiguous annotations. First, we filter to keep tables with at least one relationship between columns and have at least one column with at least 8 links[1]. Each table is then assigned to a

[1] This requirement is to help reduce ambiguity and speed up the annotation process.

Table 2. Details of the 250 WT dataset. New data is the data that is extracted from tables but is not in Wikidata.

Average number of rows	46.34
Average number of columns	5.536
% new relationships	(21235/33336) 63.7%
% new entities	(3717/21007) 17.7%
% missing entities' type	(996/21007) 4.7%
(sampled) % new relationships (after fixing entity linking (FEL))	(1464/2241) 65.3%
(sampled) % new relationships (before FEL)	(1560/2241) 69.6%
(sampled) % incorrect relationships (after FEL)	(3/2241) 0.13%
(sampled) % new or missing type entities (after FEL)	(214/1393) 15.4%
(sampled) % new or missing type entities (before FEL)	(260/1393) 18.7%

category for stratified sampling to select a maximum of 30 tables per category. The category is the most popular ontology class of the QNode's classes associated with the Wikipedia article of the table. For example, tables in Wikipedia list articles will be assigned to category `Wikimedia list article (Q13406463)`. We initially drew a sample size of 500, then two annotators annotated tables in each category one at a time (ordered by category size) until they agreed on the same semantic descriptions. However, we stopped the manual annotation process when we reached 250 tables as the cost exceeded our budget.

Table 2 shows the details of the 250 WT dataset. If we extract data from the tables using their semantic descriptions, we obtain 33,336 new relationships and 21,007 new entities or entities' types. By comparing the extracted data with Wikidata's data, we found that 63.7% and 17.7% of relationships and entities are not in Wikidata, respectively. As the comparison is computed automatically, the new data may include data that is already in Wikidata (due to errors in entity linking) or is incorrect. Therefore, we sampled 10% (24/237) of the tables that have new data to manually check and fix the linked entities, then verified the extracted relationships. We found that there are 46 (3.3%) incorrectly linked or not linked entities and only 3 (0.13%) incorrect relationships in the tables. This result shows that Wikipedia tables contain new knowledge and can be very useful to enhance Wikidata.

Finally, to compared with other systems that match tables to Wikidata, we also use a synthetic dataset from the final round of the SemTab 2020 Challenge [6]. This dataset contains approximately 22 thousand tables generated automatically from Wikidata. This dataset also comes with a list of target columns for which we need to predict the types and a list of target columns' pairs for which we need to predict the relationships. However, there are some entity columns or columns' pairs in the tables that should be annotated but are not due to not being in the target lists. Thus, for this dataset, we follow the SemTab2020 evaluation protocol to only evaluate the predictions on the items of the two lists.

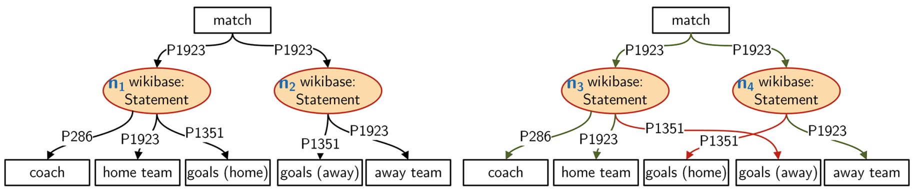


Fig. 3. Example for CPA metric (left is ground truth and right is prediction). Green and red edges are correct and incorrect, respectively. (Color figure online)

4.2 Experiment Settings

Evaluation Metrics. We assess the predicted semantic descriptions' quality in two different tasks: assigning an ontology class to a column (called an entity column) and predicting relationships in the table.

The first task is the Column-Type Annotation (CTA) task in the SemTab 2020 Challenge and is evaluated using the same metrics: approximations of precision, recall, and F_1 score. The difference between the approximate metric with its original version is the use of a scoring function, score($\cdot$), to calculate the correctness of an annotation. Let $d(x)$ be the shortest distance of the predicted class to the ground truth (GT) class. $d(x)$ is 0, 1 if the predicted class is equal to GT, or parent or child of GT, respectively. Then, $\text{score}(x) = 0.8^{d(x)}$ 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, $\text{score}(x) = 0$.

The second task is slightly different from the Column-Property Annotation (CPA) task in the SemTab 2020 challenge due to n-ary relationships. As shown in Fig. 3, despite the fact that the relationship (`P1923`, `P1351`) between `match` and `goals (home)` is the same as in the ground truth, it is not the correct relationship as it belongs to a different team. Inspired by the idea in [19], we find the best mapping between statement nodes in a predicted description to statement nodes in the ground truth description that maximizes the number of overlapping edges between them. Then, we measure the approximate precision, recall, and F_1 of edges as in the CTA task. For example, in Fig. 3, the best mapping is $\{n_3 \rightarrow n_1, n_4 \rightarrow n_2\}$ as it returns 5 overlapping edges. We have two incorrect edges: $\langle n_3,$`P1351`, `goals (away)`$\rangle$ and $\langle n_4,$`P1351`, `goals (home)`$\rangle$. Hence, the approximate precision and recall are $\frac{2}{7}$.

Baselines. We compare our method, named GRAMS, with two state-of-the-art (SOTA) systems: MantisTable [3] and BBW [18] in mapping tables to Wikidata. MantisTable achieves SOTA results on several gold-standard benchmark datasets on mapping to DBpedia. BBW is among the top-3 winners[2] of the SemTab 2020 challenge and finished in second place in the final round (within 0.1–0.2% average F1 score from the top performer). To ensure a fair assessment, we modify the

[2] We could not evaluate the other winning systems as we were unable to get access to their code and the papers do not describe them precisely.

Table 3. Performance comparison with baseline systems on CPA and CTA tasks. MantisTable* and BBW* are given correct tables' subject column.

Dataset	Method	CPA			CTA		
		Precision	Recall	F_1	Precision	Recall	F_1
250 WT	MantisTable	0.535	0.442	0.484	0.928	0.331	0.488
	MantisTable*	0.559	0.569	0.564	**0.940**	0.394	0.556
	BBW	0.796	0.123	0.214	0.850	0.233	0.367
	BBW*	0.740	0.559	0.638	0.759	0.777	0.768
	GRAMS-ST	0.526	**0.681**	0.594	–	–	–
	GRAMS	**0.824**	0.650	**0.726**	0.819	**0.813**	**0.816**
SemTab2020	MantisTable	0.985	0.976	0.981	0.977	0.800	0.880
	BBW	**0.996**	**0.995**	**0.995**	0.980	0.980	0.980
	GRAMS-ST	0.990	0.989	0.990	–	–	–
	GRAMS	**0.996**	0.994	**0.995**	**0.982**	**0.981**	**0.982**

inputs of the SOTA systems to use linked relational tables (i.e., tables' cells are already linked to entities in Wikidata) instead of plain relational tables.

In addition, we also develop another baseline, named GRAMS-ST, for comparison on the CPA task in which we replace the PSL inference with a Steiner Tree algorithm [19]. The idea of using the Steiner Tree algorithm is to find a semantic description of a table such that the total weight of relationships is minimized. The weight of a relationship is defined as the inverse of the number of rows in which we discover the relationship using Wikidata's data. Hence, it is similar to choosing the most popular relationship.

The evaluated datasets and our source code are available on Github[3].

4.3 Performance Evaluation

Table 3 shows that GRAMS outperforms the baseline systems on all tasks in all datasets, except on the CPA task of SemTab2020, where we have similar result to BBW. We report GRAMS's performance as the average of 5 independent runs (standard deviations less than 0.001) since our PSL model is a probabilistic model. In the 250 WT dataset, GRAMS exceeds the SOTA baselines by 24.2% and 32.8% of F_1 score on the CPA and CTA tasks, respectively. GRAMS also surpasses our alternative version (GRAMS-ST) by 13.2% F_1 score on the CPA task. This demonstrates that the PSL model, which takes into account both likelihood of the candidate predictions and contradicting evidence, is more robust than a model based on selecting the most frequent relationship.

The superior performance of GRAMS over the SOTA baselines on the 250 WT dataset comes from two main sources. First, MantisTable and BBW needs to identify a subject column from which we find relationships to other

[3] https://github.com/usc-isi-i2/GRAMS/releases/tag/iswc-2021.

Table 4. Average running time (seconds) per table of GRAMS in comparison with baseline systems.

Dataset	GRAMS	MantisTable	BBW
250 WT	1.155	0.627	2.674
SemTab2020	0.273	0.136	0.550

columns. Hence, their performance is significantly affected if the results of the subject column detection step are incorrect. If we give MantisTable and BBW the correct subject columns, we observe an increase in their F_1 scores on the CPA task by 8% and 42.4%, and on the CTA task by 6.8 and 40.1%, respectively. Second, tables in the 250 WT dataset are more challenging. Many tables are denormalized tables, which include more than one type of entities, require n-ary relationships or context values to model their data. Thanks to the candidate graph and the PSL model, GRAMS outperforms the SOTA baselines by 8.8% F_1 score even when they receive the correct subject columns of the tables.

However, GRAMS and the baselines do not perform well on tables that have little overlapping with Wikidata's data. For example, GRAMS can not predict correct semantic description of a table in 250 WT dataset about athletics participating in a Summer Universiade and their ranking since Wikidata do not have data of the Universiade's participation. This explains the significant gap between the F_1 score on the SemTab2020 dataset and the 250 WT dataset.

4.4 Running Time Evaluation

In this experiment, we evaluate GRAMS's running time against the baseline systems: MantisTable and BBW. The experiment is run on a single machine with Intel E5-2620v4 and 32GB RAM. We use a local key-value database to store Wikidata to avoid the network overhead in our experiment. The results are reported in Table 4. MantisTable is the fastest system and BBW is the slowest system among the three. Although our system is more complex than the baselines and is not well optimized, it has a reasonable running time especially on the SemTab2020 dataset, which contains 22127 tables. This demonstrates that our system can handle large datasets.

5 Related Work

Understanding semantics of data sources is an important task for data integration [5] and has attracted much research over the years. There are several problem formulations to address this task such as the schema matching [15] problem, which finds a correspondence between the current schema of a data source and the target schema, or semantic labeling [7,14], which assigns each attribute in a data source with one of the predefined semantic types or concepts. However, these problems are fundamentally different from the semantic modeling problem as they do not describe relationships of source attributes explicitly. Hence in the rest of this section, we will only discuss previous work that annotates both concepts and relationships of source attributes.

In general, there are two lines of research in semantic modeling, which target two different use cases. The first use case is for users who have an ontology suitable for their own problem and want to normalize their data sources according to the ontology. Methods in this line of research often take two inputs: a target ontology and a training set of known semantic descriptions. Taheriyan et al. [19] 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. As the Steiner Tree problem is NP-hard, they use an approximation algorithm to find the tree that has high frequency relationships, fewer nodes (concise), and highly overlapped with existing semantic descriptions (coherence). Vu et al. [20] developed a probabilistic graphical model (PGM) for computing the likelihood of a semantic description of a data source and use it as a scoring function to search for the most probable semantic description of a target data source. To distinguish between good and bad semantic descriptions, the PGM exploits relationships within the data and structural patterns to enforce concepts consistent with the semantic description. Despite being flexible on choosing a target ontology, these approaches suffer from the cold start problem: users need to label enough data sources before the systems can achieve good performance. This issue is more profound with a large ontology as they would need lots of training data. Thus, making these methods difficult to apply to Wikipedia tables that span many different domains.

The second use case is for harvesting structure information from millions of public web tables to publish to a knowledge graph (KG) for people to use. Generally, approaches in this line of research leverage existing knowledge in the target KG, so they are less hungry for training data. Their common methodology 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 binary relationships between columns (CPA). As the three tasks (EL, CTA, and CPA) are interdependent, Limaye et al. [10] use a probabilistic graphical model to solve them jointly. Yet, the graphical model is expensive as the number of variables in the models increases linearly with the size of the table, making it difficult to converge on an optimal solution. Mulwad et al. [11] improve it by presenting a new approximate inference algorithm named semantic message passing. However, their methods do not produce a complete semantic description as they ignore non-entity columns in the tables. Comparing to their graphical models, the size of our PSL model is not proportional to the number of rows of the tables. Our PSL model goes beyond selecting semantic description that maximizes matching scores; it incorporates structural patterns and penalizes relationships that are inconsistent with constraints in the ontology and existing knowledge in the KG. Also, PSL performs inference by solving a convex optimization problem while their approaches rely on approximate message passing algorithms.

Later work expands the problem setting to include literal columns. Ritze et al. [17] first identify a subject column of a table and candidate entities in the column, then find the candidate relationships between the subject column with other columns in the table. They iteratively update the candidate entities and candidate relationships until there is no additional change in the entity matching score with the relationship matching score. Zhang et al. [22] also use an iterative approach to 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. Nguyen et al. [12], winner of the SemTab 2019 challenge, also recalibrate the results of three tasks (EL, CTA, and CPA) after their first initial prediction. Current state-of-the-art results on the T2Dv2 and Liyame2000 standard datasets are achieved by Cremaschi et al. [3], which combine and extend features from previous work to improve the accuracy of subject column detection, and the three tasks.

Wikidata, although being popular in the Semantic Web and AI communities, is not used for the semantic modeling problem prior to the SemTab 2020 challenge. However, they do not leverage Wikidata to its full extent (e.g., qualifiers are excluded from the evaluation). New techniques used in the winning systems [2,8,13,18] of the challenge mainly depend on scoring functions to rank the matched results or fuzzy search methods to retrieve better candidate entities. In comparison to our work, most of the aforementioned methods [2,3,8,13,17,18,22] make an assumption about the table structure: a table has only one subject column, and all relationships in the table are between the subject column and other columns. This limits the ability to predict relationships between non-subject columns, which are often found in denormalized tables (e.g., two tables about books and authors are merged into one). As our approach does not make this assumption, not only can we detect relationships between non-subject columns but we also avoid the cascaded error from the subject column detection phase. Furthermore, we broaden the scope of the problem to build semantic descriptions containing n-ary relationships and implicit contextual values. Instead of using an iterative approach to solve the CTA and CPA tasks, our solution using PSL enables us to express complex dependencies between columns and their relationships and solve the tasks jointly through convex optimization. Therefore, we were able to obtain better performance than previous state-of-the-art methods.

One limitation of our approach is that it is unable to build the semantic description of a table in which each row of the table has a different property. For example, a table about awards and nominees of films has a "result" column describing whether a film won the award or not. The property `award received (P166)` should be used when the film won; otherwise, we should use the property `nominated for (P1411)`. Currently, none of the previous work on the semantic modeling problem can address this problem.

6 Conclusion and Future Work

In this paper, we present a novel graph-based approach for building semantic descriptions of Wikipedia Tables using Wikidata. Our approach constructs a candidate graph of possible relationships between columns in the table and uses collective inference to identify correct relationships and types. The evaluation shows that by using graphs to represent relationships and collective inference, our approach is robust compared to state-of-the-art systems and can handle tables with complex descriptions.

This work focuses on Wikipedia relational tables, in which we leverage existing hyperlinks. As many Web tables do not have links, we plan to extend our method to incorporate an entity disambiguation module to link cells in tables to entities in Wikidata. Another future direction of our work is to support non-relational tables by detecting layout and extracting the table data to a relational format.

We also plan to use our approach to help address the cold start problem of supervised semantic modeling systems. Specifically, we can apply our method to annotate millions of Wikipedia tables to create a large labeled dataset. This dataset can be used for weakly supervised training of semantic modeling systems on custom domain ontologies provided by the users.

Acknowledgments. This research was sponsored by the Army Research Office and the Defense Advance Research Projects Agency and was accomplished under Grant Number W911NF-18-1-0027. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office and Defense Advance Research Projects Agency or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

1. Bach, S.H., Broecheler, M., Huang, B., Getoor, L.: Hinge-loss markov random fields and probabilistic soft logic. J. Mach. Learn. Res. **18**(1), 3846–3912 (2017)
2. Chen, S., et al.: Linkingpark: an integrated approach for semantic table interpretation. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR-WS. org (2020)
3. Cremaschi, M., De Paoli, F., Rula, A., Spahiu, B.: A fully automated approach to a complete semantic table interpretation. Future Gener. Comput. Syst. **112**, 478–500 (2020)
4. Dimou, A., Sande, M.V., Colpaert, P., Verborgh, R., Mannens, E., de Walle, R.V.: Rml: a generic language for integrated rdf mappings of heterogeneous data. In: 7th Workshop on Linked Data on the Web, Proceedings, vol. 184 (2014)
5. Doan, A., Halevy, A., Ives, Z.: Principles of Data Integration. Elsevier, Edinburgh (2012)
6. Hassanzadeh, O., Efthymiou, V., Chen, J., Jiménez-Ruiz, E., Srinivas, K.: SemTab 2020: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching Data Sets (2020)
7. Hulsebos, M., 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, NY, USA (2019)

8. Huynh, V.P., Liu, J., Chabot, Y., Labbé, T., Monnin, P., Troncy, R.: Dagobah: enhanced scoring algorithms for scalable annotations of tabular data. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR-WS. org (2020)
9. Knoblock, C.A., et al.: Lessons learned in building linked data for the american art collaborative. In: ISWC 2017–16th International Semantic Web Conference (2017)
10. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. Proc. VLDB Endow. **3**(1–2), 1338–1347 (2010)
11. 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
12. Nguyen, P., Kertkeidkachorn, N., Ichise, R., Takeda, H.: Mtab: Matching tabular data to knowledge graph using probability models. CoRR abs/1910.00246 (2019)
13. Nguyen, P., Yamada, I., Kertkeidkachorn, N., Ichise, R., Takeda, H.: Mtab4wikidata at semtab 2020: tabular data annotation with wikidata. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR-WS. org (2020)
14. Pham, M., Alse, S., Knoblock, C.A., Szekely, P.: Semantic labeling: a domain-independent approach. In: Groth, P., et al. (eds.) ISWC 2016. LNCS, vol. 9981, pp. 446–462. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46523-4_27
15. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB J. **10**(4), 334–350 (2001)
16. Ritze, D., Bizer, C.: Matching web tables to dbpedia-a feature utility study. Context **42**(41), 19–31 (2017)
17. 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 (2015)
18. Shigapov, R., Zumstein, P., Kamlah, J., Oberländer, L., Mechnich, J., Schumm, I.: bbw: Matching csv to wikidata via meta-lookup. In: CEUR Workshop Proceedings, vol. 2775, pp. 17–26. RWTH (2020)
19. Taheriyan, M., Knoblock, C.A., Szekely, P., Ambite, J.L.: Learning the semantics of structured data sources. J. Web Semant. **37–38**, 152–169 (2016)
20. 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, ACM, New York, NY, USA (2019)
21. Vu, B., Pujara, J., Knoblock, C.A.: D-repr: a language for describing and mapping diversely-structured data sources to rdf. In: Proceedings of the 10th International Conference on Knowledge Capture, pp. 189–196 (2019)
22. Zhang, Z.: Effective and efficient semantic table interpretation using tableminer+. Semant. Web **8**(6), 921–957 (2017)