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

Binh Vu [a,*] and Craig A. Knoblock [a]
[a] *USC Information Sciences Institute, Marina del Rey, CA 90292, USA*
E-mails: binhvu@isi.edu, knoblock@isi.edu

Abstract. Millions of high-quality tables are 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. This paper presents a novel probabilistic approach for automatically building semantic descriptions of Wikipedia tables. Our approach leverages hyperlinks in Wikipedia tables and existing knowledge in Wikidata to construct a graph of possible relationships in a 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 a large set of Wikipedia tables by as much as 12.6% and 4.8% average F1 scores on relationship and concept prediction tasks, respectively.

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 greatly benefit many applications. DBpedia, a popular KG, has shown the success and impact of such a strategy, but only uses infoboxes. In addition to these infoboxes, Wikipedia also has millions of high-quality tables covering a wide range of domains. Leveraging these tables can help add or keep the knowledge in KGs up-to-date. For example, in an evaluation dataset collected from Wikipedia (Section 4.1), we found that approximately 64% of the relationships in the data in these tables are not present in Wikidata. Moreover, we discovered several errors in Wikidata using the tables' data. However, it is challenging to make use of these tables on a large scale because they are stored in different schemas. The task of building semantic descriptions of tables (also called semantic modeling [1, 2]) 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 directed 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 (Figure 1). From the semantic description, we can automatically generate mapping rules of mapping languages such as D-REPR [3] to convert the table data to RDF triples to import into KGs.

*Corresponding author. E-mail: binhvu@isi.edu.

Since creating semantic descriptions requires significant effort and expertise [4], many studies address this problem. Generally, they can be placed into two groups: learning-based and value-linked methods. The learning-based methods train on a set of known semantic descriptions with given domain ontologies [1, 2]. However, these methods are difficult to apply to the Wikipedia tables because little training data is available. The value-linked methods utilize KGs such as DBpedia [5, 6] or Wikidata [7, 8] to integrate data from tables. The intuition of these value-linked 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 entities.

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 that in many tables 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 since 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.

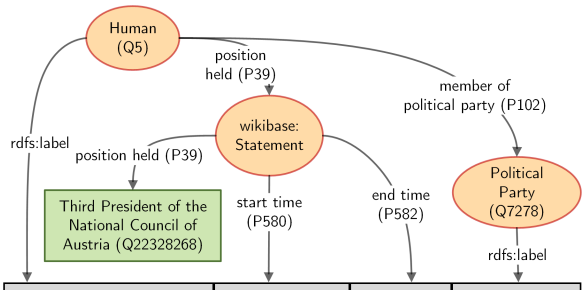
To address these issues, we present a new approach for semantic modeling that uses graphs to represent possible (n-ary) relationships in the tables and collective inference to eliminate spurious relationships on the graphs. Specifically, we construct a candidate graph containing relationships between 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) [9] model. Through collective inference, the PSL model favors edges with high confidence, is more informative, and is 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 of Wikipedia tables and on a synthetic dataset from the SemTab2020 challenge [10]. These experiments show that our method outperforms the state-of-the-art systems on the real-world dataset by 12.6% and 4.8% average F_1 scores on relationships and concept prediction tasks, respectively, while achieving similar F_1 scores on the synthetic dataset (the differences are approximately 0.1%).

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 used throughout the paper to illustrate the steps of our approach. We 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 edge `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 edge `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.



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).

In the table, some cells are linked to Wikipedia articles such as `Eva Glawischnig-Piesczek` (third row, first 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 second and third 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 first 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

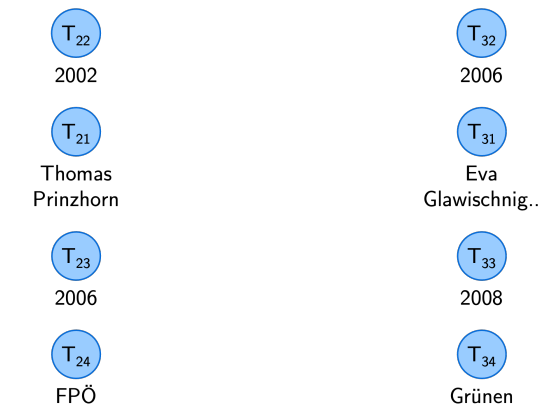
The problem of finding semantic descriptions of Wikipedia tables is defined as follows. Let T be a linked relational table, in which a cell $c_{i,j}$ of row i, column j may link to entities in a target KG, $\mathcal{C} = \{v_1, v_2, ..., v_n\}$ be a set of values (literals or entities in KG) found in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

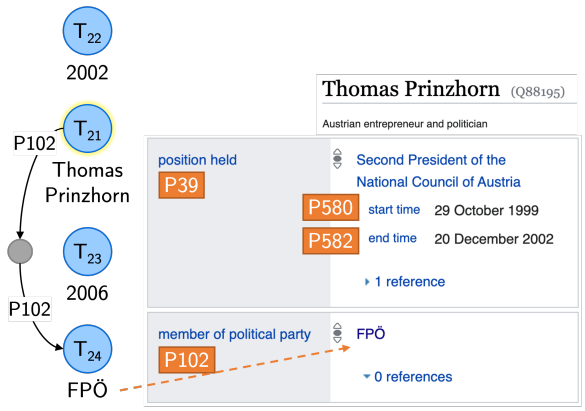
Preprocessing Since we use Wikidata as the target KG, the semantic description will be described in terms of the Wikidata ontology. Classes of the ontology are all Qnodes participating in the `subclass of (P279)` relationship, and properties of the ontology are all Wikidata properties and the `rdfs:label`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

3.1. Constructing Candidate Graphs

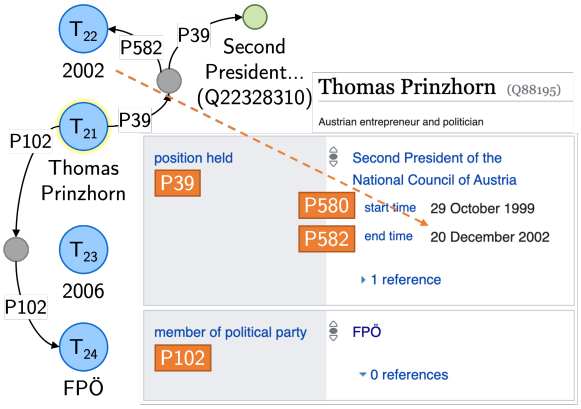
To create a candidate graph, we first create a data graph of all possible relationships between table cells and table context. Then we summarize the data graph to obtain relationships between columns and table context.



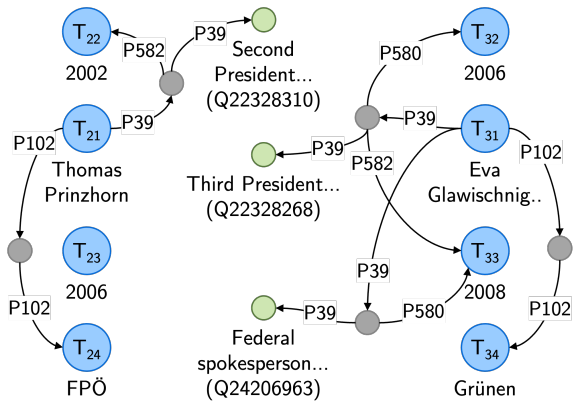
(a) The data graph is first initialized with table cells.



(b) Added the relationship P102 between cells T_{21} and T_{24} as found in Wikidata.



(c) Cell T_{22} is matched to the value of the qualifier P582 of the statement P39. To make the statement valid, the value of the statement, which is the entity Second President, is also added to the graph.



(d) An excerpt of the final data graph.

Fig. 2. Illustrations of steps in building the data graph of the table in Figure 1 using Algorithm 1. Edges are displayed without their full labels for readability. Grey, blue, and green nodes are statements, table cells, and entities, respectively.

Algorithm 1: CONSTRUCT DATA GRAPH

Output: the data graph G_d
Input: Input table T, input context $\mathcal{C} = \{v_1, ..., v_n\}$, max hop maxHop

$G_d \leftarrow$ empty graph
add cells in the table ($c_{i,j} \in T$.cells) as nodes to G_d
add values in the context ($v_i \in \mathcal{C}$) as nodes to G_d
for $n_i \leftarrow G_d$.nodes **do**
  for $n_j \leftarrow G_d$.nodes **do**
    if CanLink (n_i, n_j) **then**
      for $e_i \leftarrow n_i$.linkedEntities **do**
        for $e_j \leftarrow n_j$.linkedEntities **do**
          add FindEnt2EntPaths (e_i, e_j, maxHop) to G_d
        add FindEnt2LiteralPaths (e_i, n_j) to G_d
InferAdditionalLinks (G_d)
return G_d

Constructing Data Graphs Algorithm 1 outlines the process of building a data graph, which is also illustrated in Figure 2. To begin, we add cells of T and items in $\mathcal{C}$ as nodes to an empty graph G_d (lines 1 - 3, Figure 2(a)). Then we find paths in Wikidata that connect two nodes in G_d using two functions `FindEnt2EntPaths` and `FindEnt2LiteralPaths` (lines 4 - 10). The former function simply returns paths between two entities in Wikidata (Figure 2(b)). The latter function returns paths from an entity to a literal (Figure 2(c)). 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 `maxHop` 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 (Figure 2(c)). Note that we only need to find paths between pairs of nodes that can be linked (line 6). 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.

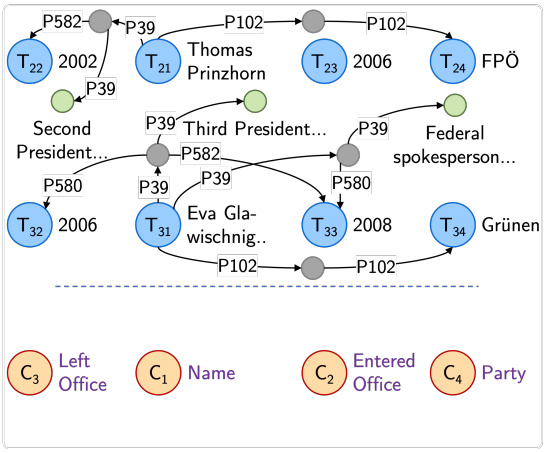
The discovered paths are then added to G_d such that the original identifiers of Wikidata statements and entities are preserved (lines 9 and 10). This allows paths of n-ary relationships to be connected automatically as they share the same Wikidata statements. Figure 2(d) shows an excerpt of the data graph of the table in the motivating example after this step.

Finally, we infer additional edges in G_d based on logical rules (inverse, sub-property, and transitive rules) specified in the Wikidata ontology (line 13). The intuition is that the final graph G_d after inference should be the same as if we run inference on KG, then build the data graph G_d. For example, if we have a link `capital (P36)` from node `France` to node `Paris`, we add a link `capital of (P1376)` from `Paris` to `France` because `P1376` is an inverse property of `P36`. Similarly, we add an edge `location (P276)` between nodes u_d and v_d in the graph G_d if there is an edge `located in admin... (P131)` between them as `P131` is a sub-property of `P276`. Note that for the sub-property rule, we only consider parent properties found in G_d to avoid generating unnecessary top-level properties such as `rdf:Property` and to keep a reasonable size of G_d.

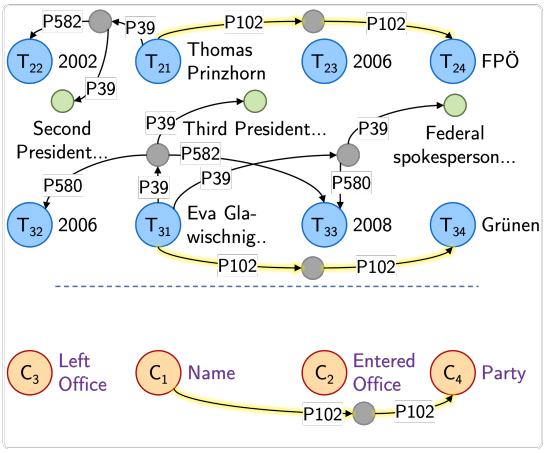
Algorithm 2: CONSTRUCT CANDIDATE GRAPH

Input: A data graph G_d, Input table T
Output: the candidate graph G_s

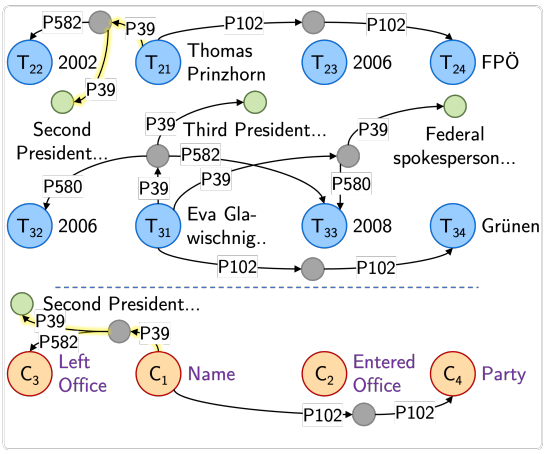
1 $G_s \leftarrow$ empty graph
2 add columns in the table T as nodes to G_s
3 add literal or entity nodes in G_d as nodes to G_s, keeping their original id
4 **for** $u_d \in G_d$.nodes **do**
5 **for** $v_d \in G_d$.nodes **do**
6 **if** v_d is value of a property e of u_d **then**
7 $\mathsf{stmt}_d \leftarrow$ 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 $\mathsf{stmt}_s \leftarrow$ 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: $u_s \xrightarrow{e} \mathsf{stmt}_s \xrightarrow{e} v_s$
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 $\mathsf{stmt}_s \xrightarrow{q} t_s$ does not exist **then**
16 add qualifier $\mathsf{stmt}_s \xrightarrow{q} t_s$ to G_s
17 **return** G_s



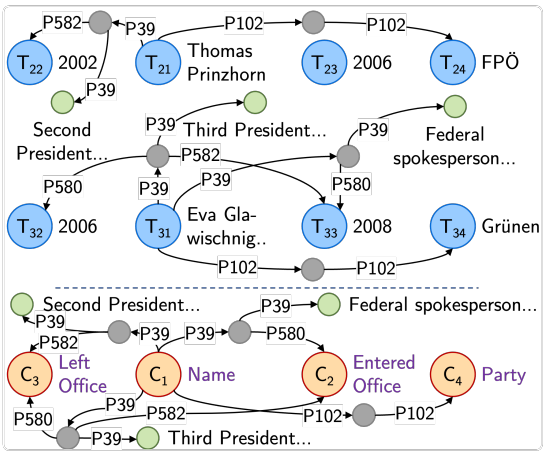
(a) The candidate graph (under the dash line) is first initialized with table columns.



(b) The relationships P102 between cells of columns 1 and 2 are grouped and added to the candidate graph.



(c) The relationship P39 between cells in column 1 and the entity Second President is added to the candidate graph. Then, the qualifier P582 to column 3 is added to the graph.



(d) An excerpt of the final candidate graph (under the dash line)

Fig. 3. Illustration of steps in building the candidate graph of the table in Figure 1 using Algorithm 2. The graphs on top of the dash lines are the data graphs; the graphs under are the candidate graphs. Edges are displayed without their full labels for readability. Grey, blue, orange, and green nodes are statements, table cells, table columns, and entities, respectively.

Constructing Candidate Graphs With the data graph G_d built from the previous step, we 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 Figure 3(b), 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 and is illustrated in Figure 3. It starts by adding columns in the tables as nodes to G_s (lines 1 and 2). 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 (lines 12 - 16).

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 [9]. 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:

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

where w is weight of the rule, CLOSEFRIEND, FRIEND are predicates, A, B, C are variables. A ground predicate can have a value between 0 and 1 (e.g., $\textsc{CloseFriend}(\text{"James"}, \text{"Mary"}) = 0.8$), and is observed if its value is known. 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 observations (ground predicates whose values are known), PSL substitutes (grounds) predicates in the rules with the observations and performs convex optimization to infer the values of the unobserved predicates.

3.2.1. PSL model

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

$$\neg\textsc{CorrectRel}(U, V, S, P) \tag{1}$$

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

$$\textsc{Rel}(U, V, S, P) \wedge \textsc{PosRelFeat}_i(U, V, S, P) \Rightarrow \textsc{CorrectRel}(U, V, S, P) \tag{3}$$

$$\textsc{Rel}(U, V, S, P) \wedge \textsc{NegRelFeat}_i(U, V, S, P) \Rightarrow \neg\textsc{CorrectRel}(U, V, S, P) \tag{4}$$

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

$$\textsc{Rel}(U, V, S_1, P_1) \wedge \textsc{Rel}(U, V, S_2, P_2) \wedge (S_1 != S_2) \wedge \textsc{SubProp}(P_1, P_2) \Rightarrow \neg\textsc{CorrectRel}(U, V, S_2, P_2) \tag{6}$$

$$\textsc{Type}(U, T) \wedge \textsc{TypeDistance}(U, T) \Rightarrow \neg\textsc{CorrectType}(U, T) \tag{7}$$

$$\textsc{CorrectRel}(U, V, S, P) \wedge \textsc{StatementProperty}(S, P) \wedge \neg\textsc{Domain}(P, T) \Rightarrow \neg\textsc{CorrectType}(U, T) \tag{8}$$

$$\textsc{CorrectRel}(U, V, S, P) \wedge \neg\textsc{Range}(P, T) \Rightarrow \neg\textsc{CorrectType}(V, T) \tag{9}$$

$$\textsc{CorrectRel}(U, V, S, P) \wedge \textsc{DatatypeProperty}(P) \Rightarrow \neg\textsc{CorrectType}(V, T) \tag{10}$$

$$\textsc{CorrectRel}(U, V, S, P_1) \wedge \textsc{Rel}(V, T, S_2, P_2) \wedge \textsc{OneToMany}(V, T) \Rightarrow \neg\textsc{CorrectRel}(V, T, S_2, P_2) \tag{11}$$

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

Unlike the previous rules, the remaining rules are applied to more than one relationship or type. They are used to enforce consistency of the descriptions with the Wikidata data model as well as to introduce prior knowledge of the desired semantic descriptions. Specifically, rules 6 and 7 favor fine-grained properties and types. Rules 8 and 9

Table 1
Predicates in the PSL model.

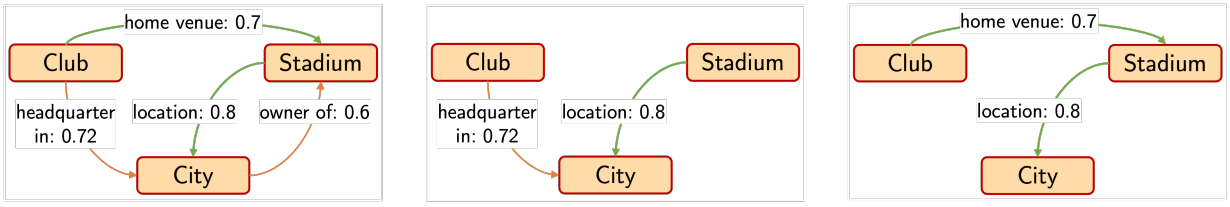
Predicates	Meaning
$\text{REL}(U, V, S, P)$	A candidate relationship P of statement S between nodes U and V
$\text{CORRECTREL}(U, V, S, P)$	Denoting if a relationship $\text{REL}(U, V, S, P)$ is correct
$\text{TYPE}(U, T)$	A candidate type T of column N
$\text{CORRECTTYPE}(U, T)$	Denoting if a type $\text{TYPE}(U, T)$ is correct
$\text{SUBPROP}(P_1, P_2)$	Property P_1 is a subproperty of P_2
$\text{STATEMENTPROPERTY}(S, P)$	P is the main property of statement S
$\text{POSRELFEAT}_i(U, V, S, P)$	Value of feature i backing the relationship $\text{REL}(U, V, S, P)$
$\text{NEGRELFEAT}_i(U, V, S, P)$	Value of feature i opposing the relationship $\text{REL}(U, S, V, P)$
$\text{POSTYPEFEAT}_i(U, T)$	Value of feature i backing the column type (U, T)
$\text{DOMAIN}(P, T)$	Type T is a domain of property P
$\text{RANGE}(P, T)$	Type T is a range of property P
$\text{DATATYPEPROPERTY}(P)$	Value of property P is a literal data (e.g., numbers, strings, datetimes)
$\text{ONETOMANY}(U, V)$	A value in column U is associated with multiple values in column V
$\text{TYPEDISTANCE}(U, T)$	The shortest distance of type T of column U to the most fine-grained candidate types of U divided by the longest distance

ensure that the predicted type of a column should be one of the domains and ranges of the outgoing and incoming ontology property, respectively. In addition, if an ontology property is a datatype property, the target of the property cannot be instances of a class (rule 10). Finally, rule 11 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 the PSL tutorial ($w = 1$); rules that introduce preferences should have very small weights (rules 6 and 7 have $w = 0.1$); and rules that act as constraints should have very high weights (rule 11 has $w = 100$).

3.2.2. *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 edges 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.



(a) Pruned graph with edges having probabilities greater than or equal to a threshold.

(b) Selecting edges with the highest probabilities missed a relationship between Club and Stadium.

(c) Selecting a tree that has the highest average probabilities.

Fig. 4. An example of relationships with their probabilities predicted by PSL. Green edges are correct while red edges are incorrect.

The type with the highest probability for each column is selected. However, using the same approach for selecting relationships is not ideal as it can result in multiple paths between two nodes. In many cases, a single path is sufficient to describe the relationship between two nodes. Including extra paths, which tends to be incorrect, can decrease the precision of our prediction. For example, Figure 4(a) shows four remaining relationships (after pruning

Algorithm 3: FIND STEINER TREE

Input: - G_{sp}: a candidate graph that its low probability edges were removed
- K_{path}: keeping maximum K paths that reach a node
- K_{tree}: keeping maximum K trees

Output: A predicted semantic description

```
1  if | G_sp.connectedComponents | > 1 then
2      tree ← empty tree
3      for component ∈ G_sp.connectedComponents do
4          recursive call the algorithm with the component and merge the result into tree
5      return tree
6  Add a pseudo root node to G_sp and edges from the pseudo root node to all nodes in G_sp
7  pathsBetweenNodes ← {};
8  for columnNode ∈ G_sp.nodes do
9      for node ∈ G_sp.nodes do
10         pathsBetweenNodes [columnNode, node ] ← top K_path from node to columnNode
11 roots ← list of nodes that are all visited by all column nodes and not statement nodes
12 bestTree ← null
13 for root ∈ roots do
14     topKTrees ← []
15     for columnNode ∈ G_sp do
16         if topKTrees is empty then
17             topKTrees ← create a tree for each path of pathsBetweenNodes [columnNode, root ]
18             continue;
19         nextTrees ← set()
20         for path, tree ∈ pathsBetweenNodes [columnNode, root ] × topKTrees do
21             newTree ← merge path into tree
22             fix statement nodes in newTree that do not have their property
23             add newTree to nextTrees
24         topKTrees ← K_tree best trees in nextTrees
25     bestTree ← best tree of topKTrees and bestTree
26 Remove the pseudo root node from bestTree if it exists
27 Add context nodes that are correct to bestTree
28 return bestTree
```

ones with low probabilities) in which only two are correct. In addition, it is rare to have two or more independent main subjects, which are nodes that do not have any incoming relationships, in a table. This is demonstrated in the example in Figure 4(b). In other words, there often exists a path between any pair of nodes in a semantic description. For these reasons, our goal is to select relationships between columns and entity/literal nodes that form a directed rooted tree with the highest average probabilities of edges. The tree may contains other nodes, hence it is a directed Steiner tree.

Algorithm 3 presents the pseudo code of this step. Inspired from the backward search algorithm [11], the idea is identifying a root node that can reach all column nodes, then generating trees by merging paths from the root node to the column nodes. Lines 8 to 10 finds the top K paths between a pair of node in the graph similar to finding K shortest path. Then, for each potential root node, we merge the paths from the root node to each column node in round robin fashion (lines 14 to 24). If the tree after merging contains statement nodes that do not have their property

Table 2
Details of the 250WT 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%

(i.e., only have qualifiers) and the statement property is in the graph, we add the statement property back (line 21). Finally, to ensure that we always have at least one root node, we add a pseudo node that has edges to all remaining nodes, then remove the pseudo node after the algorithm finds the Steiner tree (lines 10 and 25). The weight of edges of the pseudo node is the total weight of edges in the graph plus 1 so that we do not use those edges unless there is no common root node.

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 [12] or Limaye [13]. 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 250WT[1], with their semantic descriptions built using the Wikidata ontology.

The new dataset's tables are selected from a pool of approximately 2,000,000 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[2]. Each table is then assigned to a 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 250WT 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 compare with other systems that match tables to Wikidata, we also use a synthetic dataset from the final round of the SemTab 2020 Challenge [10]. This dataset contains approximately 22,000 tables generated au-

[1] https://github.com/binh-vu/sm-datasets
[2] This requirement is to help reduce ambiguity and speed the annotation process.

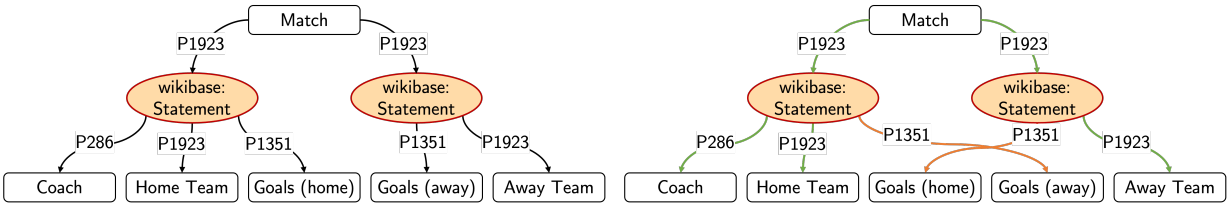


Fig. 5. Example for CPA metrics (left is ground truth and right is prediction). Green and red edges are correct and incorrect, respectively.

tomatically 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.

4.2. Evaluation Tasks

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; the performance on each table is measured by approximations of precision and recall. The difference between the approximate metric and its original version is the use of a scoring function, $\text{score}(\cdot)$, to calculate the correctness of an annotation. Specifically, let $d(x)$ be the shortest distance of the predicted class to the ground truth (GT) class where $d(x)$ is 0 if the predicted class is equal to GT, is 1 if the predicted class is the parent or child of GT. Then $\text{score}(x) = 0.8^{d(x)}$ if $d(x) \leqslant 5$ and x is a correct annotation or an ancestor of GT; $\text{score}(x) = 0.7^{d(x)}$ if $d(x) \leqslant 3$ 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 Figure 5, 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 because it belongs to a different team. Inspired by the idea in [1], 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 Figure 5, 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: $\langle n_3, \texttt{P1351}, \texttt{goals (away)}\rangle$ and $\langle n_4, \texttt{P1351}, \texttt{goals (home)}\rangle$. Hence, the approximate precision and recall are $\frac{2}{7}$.

We report the results of the tasks as the average of 10 independent runs. For readability, we omit $\pm 0.1\%$ in our tables when the standard errors of mean (SE) are less than 0.1% unless otherwise specified.

4.3. Evaluation Baselines

We compare our method, GRAMS, with three state-of-the-art (SOTA) systems: MantisTable [5], BBW [8], and MTab [7] in mapping tables to Wikidata. MantisTable achieves SOTA results on several gold-standard benchmark datasets on mapping to DBpedia. BBW is among the top three winners 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). MTab is the winner of the SemTab 2020 challenge. To ensure a fair assessment, we modify the inputs of the SOTA systems to use linked relational tables (i.e., tables' cells are already linked to entities in Wikidata) instead of unlinked relational tables.

In addition, we also develop another baseline, GRAMS-ST, for comparison on the CPA task in which we replace the PSL inference with the Steiner tree algorithm (Algorithm 3). 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.

Table 3
Performance comparison with baseline systems on CPA and CTA tasks. MantisTable†, BBW†, and MTab† are given correct tables' subject column. *SE = ±0.2%

Dataset	Method	CPA			CTA		
		Avg Precision	Avg Recall	Avg F_1	Avg Precision	Avg Recall	Avg F_1
250WT	MantisTable	53.0%	44.7%	48.5%	**88.4%**	30.4%	45.2%
	MantisTable†	55.0%	57.1%	56.0%	87.7%*	34.9%	49.9%
	BBW	75.6%	11.8%	20.5%	83.1%	23.0%	36.0%
	BBW†	63.6%	58.9%	61.2%	73.9%	76.1%	75.0%
	MTab	83.9%	48.5%	61.5%	77.0%	77.0%	77.0%
	MTab†	84.8%*	54.6%	66.4%	77.5%	77.5%	77.5%
	GRAMS-ST	58.6%	**70.9%**	64.2%	-	-	-
	GRAMS	**88.1%**	63.9%	**74.1%**	81.3%	**82.4%**	**81.8%**
SemTab2020	MantisTable	98.3%	97.5%	97.9%	95.9%	79.0%	86.6%
	BBW	99.2%	**99.2%**	99.2%	96.7%	96.7%	96.7%
	MTab	**99.5%**	99.1%	**99.3%**	**97.1%**	**97.1%**	**97.1%**
	GRAMS-ST	99.2%	99.1%	99.2%	-	-	-
	GRAMS	99.3%	99.1%	99.2%	97.0%	97.0%	97.0%

Our source code, baselines, and evaluated data are available on Github[3].

4.4. Performance Evaluation

Table 3 shows that GRAMS outperforms the baseline systems on all tasks on the 250WT dataset and has similar average F_1 scores to the SOTA baseline on the SemTab2020 dataset with the differences are approximately 0.1%. On the 250WT dataset, GRAMS exceeds the SOTA baseline by 12.6% and 4.8% F_1 scores on the CPA and CTA tasks, respectively. GRAMS also surpasses our alternative version (GRAMS-ST) by 9.9% F_1 score on the CPA task. This demonstrates that the PSL model, which takes into account both the 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 250WT dataset comes from two main sources. First, MantisTable, BBW, and MTab need to identify a subject column from which we find relationships to other 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 7.5% and 40.7%, and on the CTA task by 4.7% and 39.0%, respectively. Second, tables in the 250WT 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 7.7% F_1 score (CPA) and 4.3% F_1 score (CTA) even when the baselines receive the correct subject columns of the tables.

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

4.5. Table Subject Detection

In this experiment, we evaluate the role of context in GRAMS in helping to recognize the correct subject of the tables. We consider that the subject of a table is what the table is primarily about. It is often expressed in the

[3]https://github.com/usc-isi-i2/GRAMS/releases/tag/swe22

semantic description via ontology classes (e.g., a table about districts of Uzbekistan uses the class `districts of Uzbekistan` for a column `districts`) or relationships with literal values. For example, the subject of the example table is politicians who are `Third President of National Council... (Q22328268)` (expressed via property `position held (P39)`).

We report the performance of GRAMS on the 250WT dataset because it annotates the table subjects. Overall, 76 tables have subjects and GRAMS can predict the subjects of 18 (23.68%). Among the 18 tables, 17 are predicted correctly and one table is not predicted exactly as the annotated ground truth. The table is about locations of an event that GRAMS uses the property `destination point (P1444)` instead of the property `location (P276)` used in the gold annotation. We consider that it is still semantically correct because `P1444` is a direct subproperty of `P276` and is actually used to express the locations of that event in Wikidata. Without using context, GRAMS does not predict the subjects; hence, the recall is 0. We do not include the baselines in this experiment because they do not predict the table subjects.

Analyzing the remaining tables that GRAMS cannot predict the subjects, we find that 48 tables (63.16%) do not have any overlapping data with their subjects in Wikidata and 10 tables (13.16%) have some overlapping data. Among the 48 tables that have no overlapping data, 17 (22.37%) of them have relationships with the subjects that can be derived based on the time range. For example, the table subject is about politicians serving in the 24th Parliament of British Columbia, and some politicians are members of Legislative Assembly of British Columbia from 1952 to 1972. However, such inference can be noisy and may require specific knowledge of the ontology; hence, we leave this for future work. For the 10 tables having some overlapping data but GRAMS cannot predict, the main reason is that besides the overlapping data, Wikidata also has data of the subjects that are different than the data in the tables. For example, in a table about movies that an actor/actress is a cast member of, a few movies list them in their casting list but other movies in the table do not; they only list their lead actors/actresses. Because of that, our PSL model considers the matched relationship coincident and discard the correct subject.

4.6. Feature Analysis and Post-processing Evaluation

To understand the impact of the PSL rules on the result, we perform ablation analysis by removing each of three groups of rules: fine-grained properties and classes (rules 6, 7), relationship-type agreement (rules 8, 9, 10), and functional dependency (rule 11). The results are reported in Table 4. The most important group is fine-grained properties/classes. Removing this group decreases the performance of both CPA and CTA tasks. This is expected because Wikidata ontology is deep and utilizes specific properties and classes to express different nuances. For example, at the time of writing, an entity such as United States (Q30) is associated with 8 different types. The function dependency group mainly affects the precision of relationship predictions as removing it decreases the CPA precision by 2.5% while the CPA recall is roughly the same. The CTA scores slightly drop due to less accurate relationship predictions. Finally, the relationship-type agreement group has no apparent effect on the average scores on both tasks on 250WT. Because the automatic entity linking step is highly accurate, the domains and ranges of the discovered candidate relationships mostly agree with the corresponding candidate types. Thus this explains the little effect of these rules. We conjecture that this group will be more important when the quality of the entity linking decreases.

Table 4

Ablation analysis of the groups of rules in GRAMS. Each cell value is a difference in a score when dropping one group of rules.

	CPA			CTA		
	Avg Precision	Avg Recall	Avg F_1	Avg Precision	Avg Recall	Avg F_1
GRAMS	**88.1**%	**63.9**%	**74.1**%	**81.3**%	82.4%	81.8%
− Fine-grained Properties and Classes (FG)	86.4%	62.7%	72.7%	80.7%	81.8%	81.3%
− Functional Dependency (FD)	85.6%	63.9%	73.2%	80.9%	82.1%	81.5%
− Relationship-Type Agreement (RT)	88.0%	63.8%	74.0%	81.3%	**82.4**%	**81.9**%
− FG − RT − FD	84.1%	63.0%	72.0%	80.8%	82.1%	81.5%

Table 5 shows the performance of our post-processing algorithm (Steiner tree) in Section 3.2.2 versus two baselines: selecting relationships with the highest scores between two nodes (named Pairwise Selection) and finding the minimum arborescence (named Minimum Arborescence). The first baseline has a slightly higher CPA recall than all methods since it may predict more than one relationship per node. However, because of that, it has the lowest CPA precision. The second baseline has better performance than the first baseline since it enforces relationships to form a tree similar to our Steiner tree method. However, it has smaller scores than the Steiner tree method. The reason is that entity/literal nodes are optional in the Steiner tree method and only used if they are needed to describe the relationships (e.g., connecting two nodes).

Table 5
Performance of GRAMS with respected to different post-processing algorithms on the 250WT dataset.

	CPA			CTA		
	Avg Precision	Avg Recall	Avg F_1	Avg Precision	Avg Recall	Avg F_1
Pairwise Selection	79.8%	**65.5%**	71.9%	80.5%	**82.6%**	81.6%
Minimum Arborescence	86.9%	63.4%	73.3%	81.3%	82.3%	81.8%
Steiner Tree	**88.1%**	63.9%	**74.1%**	**81.3%**	82.4%	**81.8%**

4.7. Running Time Evaluation

In this experiment, we evaluate GRAMS's running time against the baseline systems: MantisTable, BBW, and MTab. The experiment is run on a single laptop with M1 Pro and 32GB RAM. We use a local key-value database to store Wikidata to avoid network overhead. The results are reported in Figure 6. The baseline systems have much shorter running times on the SemTab2020 dataset than on the 250WT dataset. The reason is that on the SemTab2020 dataset, the targets of CPA (i.e., pairs of columns) and CTA (i.e., columns) tasks are provided, while on the 250WT dataset, the targets are unknown and the baselines have to find them. On the other hand, GRAMS shows similar running times on both datasets because our implementation does not rely on the given CPA and CTA targets. Although our system is more complex than the baselines and is not well optimized, it has a reasonable running time of approximately 0.2 seconds per table, and is faster than MTab and BBW on the 250WT dataset.

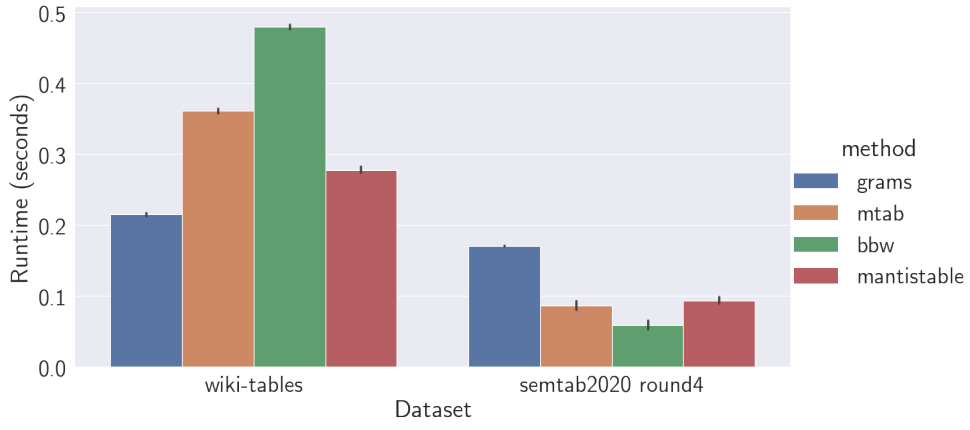


Fig. 6. Average running time (seconds) per table of GRAMS in comparison with the baseline systems.

5. Related Work

Understanding the semantics of data sources is an important task in data integration [14]. Several problem formulations 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, and semantic labeling [16, 17], 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 because they do not describe relationships of source attributes explicitly. Therefore we 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. [1] 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. Because the Steiner tree problem is NP-hard, they use an approximation algorithm to find the tree with high frequency relationships, with fewer nodes (concise), and that is highly overlapped with existing semantic descriptions (coherence). Vu et al. [2] 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 conceptual consistency within the semantic description. Despite being flexible in choosing a target ontology, these approaches suffer from the cold start problem: users need to label sufficient data sources for the systems to achieve good performance. This issue is more profound with large ontologies because they need a large amount of training data, 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 in order to publish to a knowledge graph (KG) for public 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). Because the three tasks (EL, CTA, and CPA) are interdependent, Limaye et al. [13] use a probabilistic graphical model to solve them jointly. Yet the graphical model is expensive because 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. [18] improve the probabalistic graphical model by presenting a new approximate inference algorithm named semantic message passing. However, their methods do not produce a complete semantic description because they ignore non-entity columns in the tables. Compared 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 a 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. [12] first identify a subject column of a table and candidate entities in the column, then find the candidate relationships between the subject column and 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. [19] 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. [20], 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. [5], which combine and extend features from previous work to improve the accuracy of subject column detection and the three tasks.

Wikidata, although popular in the Semantic Web and AI communities, is not used for the semantic modeling problem before the SemTab 2020 challenge. However, in the challenge, they do not leverage Wikidata to its full extent (e.g., qualifiers are excluded from the evaluation). New techniques used in the challenge's winning systems [7,

8, 21, 22] 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 [5, 7, 8, 12, 19, 21, 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). Because 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 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. Because 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 the weakly-supervised training of semantic modeling systems on custom domain ontologies provided by users.

References

[1] M. Taheriyan, C.A. Knoblock, P. Szekely and J.L. Ambite, Learning the semantics of structured data sources, *Journal of Web Semantics* **37-38** (2016), 152–169.

[2] B. Vu, C. Knoblock and J. Pujara, Learning Semantic Models of Data Sources Using Probabilistic Graphical Models, in: *The World Wide Web Conference*, WWW '19, ACM, New York, NY, USA, 2019, pp. 1944–1953. ISBN 978-1-4503-6674-8.

[3] B. Vu, J. Pujara and C.A. Knoblock, D-REPR: A Language for Describing and Mapping Diversely-Structured Data Sources to RDF, in: *Proceedings of the 10th International Conference on Knowledge Capture*, 2019, pp. 189–196.

[4] C.A. Knoblock, P. Szekely, E. Fink, D.N. Duane Degler, R. Sanderson, K. Blanch, S. Snyder, N. Chheda, N. Jain, R.R. Krishna, N.B. Sreekanth and Y. Yao, Lessons Learned in Building Linked Data for the American Art Collaborative, in: *ISWC 2017 - 16th International Semantic Web Conference*, 2017.

[5] M. Cremaschi, F. De Paoli, A. Rula and B. Spahiu, A fully automated approach to a complete Semantic Table Interpretation, *Future Generation Computer Systems* **112** (2020), 478–500.

[6] D. Ritze and C. Bizer, Matching web tables to dbpedia-a feature utility study, *context* **42**(41) (2017), 19–31.

[7] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise and H. Takeda, Mtab4wikidata at semtab 2020: Tabular data annotation with wikidata, *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR-WS. org* (2020).

[8] R. Shigapov, P. Zumstein, J. Kamlah, L. Oberländer, J. Mechnich and I. Schumm, bbw: Matching CSV to Wikidata via Meta-lookup, in: *CEUR Workshop Proceedings*, Vol. 2775, RWTH, 2020, pp. 17–26.

[9] S.H. Bach, M. Broecheler, B. Huang and L. Getoor, Hinge-Loss Markov Random Fields and Probabilistic Soft Logic, *J. Mach. Learn. Res.* **18**(1) (2017), 3846–3912–.
[10] O. Hassanzadeh, V. Efthymiou, J. Chen, E. Jiménez-Ruiz and K. Srinivas, SemTab 2020: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching Data Sets, Zenodo, 2020.
[11] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti and S. Sudarshan, Keyword searching and browsing in databases using BANKS, in: *Proceedings 18th International Conference on Data Engineering*, 2002, pp. 431–440.
[12] D. Ritze, O. Lehmberg and C. Bizer, Matching html tables to dbpedia, in: *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, 2015, pp. 1–6.
[13] G. Limaye, S. Sarawagi and S. Chakrabarti, Annotating and Searching Web Tables Using Entities, Types and Relationships, *Proc. VLDB Endow.* **3**(1–2) (2010), 1338–1347.
[14] A. Doan, A. Halevy and Z. Ives, *Principles of data integration*, Elsevier, 2012.
[15] E. Rahm and P.A. Bernstein, A survey of approaches to automatic schema matching, *the VLDB Journal* **10**(4) (2001), 334–350.
[16] M. Pham, S. Alse, C. Knoblock and P. Szekely, Semantic labeling: A domain-independent approach, in: *ISWC 2016 - 15th International Semantic Web Conference*, 2016.
[17] M. Hulsebos, K. Hu, M. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, c. Demiralp and C. Hidalgo, Sherlock: A Deep Learning Approach to Semantic Data Type Detection, in: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 1500–1508–. ISBN 9781450362016.
[18] V. Mulwad, T. Finin and A. Joshi, Semantic message passing for generating linked data from tables, in: *International Semantic Web Conference*, Springer, 2013, pp. 363–378.
[19] Z. Zhang, Effective and efficient semantic table interpretation using tableminer+, *Semantic Web* **8**(6) (2017), 921–957.
[20] P. Nguyen, N. Kertkeidkachorn, R. Ichise and H. Takeda, MTab: Matching Tabular Data to Knowledge Graph using Probability Models, *CoRR* **abs/1910.00246** (2019).
[21] S. Chen, A. Karaoglu, C. Negreanu, T. Ma, J.-G. Yao, J. Williams, A. Gordon and C.-Y. Lin, Linkingpark: An integrated approach for semantic table interpretation, *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR-WS. org* (2020).
[22] V.-P. Huynh, J. Liu, Y. Chabot, T. Labbé, P. Monnin and R. Troncy, DAGOBAH: Enhanced Scoring Algorithms for Scalable Annotations of Tabular Data, *Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR-WS. org* (2020).
[23] A. Dimou, M.V. Sande, P. Colpaert, R. Verborgh, E. Mannens and R.V. de Walle, RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data, *7th Workshop on Linked Data on the Web, Proceedings* **1184** (2014).
[24] V. Efthymiou, O. Hassanzadeh, M. Rodriguez-Muro and V. Christophides, Matching Web Tables with Knowledge Base Entities: From Entity Lookups to Entity Embeddings, 2017, pp. 260–277. ISBN 978-3-319-68287-7.
[25] E. Jimenez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas and V. Cutrona, Results of semtab 2020, in: *CEUR Workshop Proceedings*, Vol. 2775, 2020, pp. 1–8.
[26] A. Kimmig, A. Memory, R.J. Miller and L. Getoor, A Collective, Probabilistic Approach to Schema Mapping Using Diverse Noisy Evidence, *IEEE Transactions on Knowledge and Data Engineering* (2018).
[27] D. Zhang, M. Hulsebos, Y. Suhara, c. Demiralp, J. Li and W.-C. Tan, Sato: Contextual Semantic Type Detection in Tables, *Proc. VLDB Endow.* **13**(12) (2020), 1835–1848–.
[28] J. Chen, E. Jiménez-Ruiz, I. Horrocks and C. Sutton, Colnet: Embedding the semantics of web tables for column type prediction, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33, 2019, pp. 29–36.