Radar Station: Using KG Embeddings for Semantic Table Interpretation and Entity Disambiguation

Jixiong Liu[1,2][0000−0002−8750−8637], Viet-Phi Huynh[1], Yoan Chabot[1][0000−0001−5639−1504], and Raphael Troncy[2][0000−0003−0457−1436]

[1] Orange, Belfort, France
[2] EURECOM, Sophia-antipolis, France
yoan.chabot@orange.com

Abstract. Relational tables are widely used to store information about entities and their attributes and they are the de-facto format for training AI algorithms. Numerous Semantic Table Interpretation approaches have been proposed in particular for the so-called cell-entity annotation task aiming at disambiguating the values of table cells given reference knowledge graphs (KGs). Among these methods, heuristic-based ones have demonstrated to be the ones reaching the best performance, often relying on the column types and on the inter-column relationships aggregated by voting strategies. However, they often ignore other column-wised semantic similarities and are very sensitive to error propagation (e.g. if the type annotation is incorrect, often such systems propagate the entity annotation error in the target column). In this paper, we propose Radar Station, a hybrid system that aims to add a semantic disambiguation step after a previously identified cell-entity annotation. Radar Station takes into account the entire column as context and uses graph embeddings to capture latent relationships between entities to improve their disambiguation. We evaluate Radar Station using several graph embedding models belonging to different families on Web tables as well as on synthetic datasets. We demonstrate that our approach can lead to an accuracy improvement of 3% compared to the heuristics-based systems. Furthermore, we empirically observe that among the various graph embeddings families, the ones relying on fine-tuned translation distance show superior performance compared to other models.

Keywords: Cell-entity annotation · Graph Embeddings · Semantic Table Interpretation · Entity Disambiguation.

1 Introduction

Tabular data is one of the most commonly used formats. This condensed representation of information offers a compact visualisation of the data that is easy for users to access and use. Among the wide variety of tabular data, relational tables organise entity attributes into columns and are used extensively in enterprise data repositories and on the Web for storing information and for training

AI algorithms. We argue that adding a semantic layer on top of such rich data source using KGs can be beneficial to several downstream tasks such as datasets indexing [2], KG enrichment [25], or dataset recommendation [32]. This process of automatically understanding what tabular data is about is named Semantic Table Interpretation (STI). Cell-entity annotation [16] (CEA) is one of the fundamental tasks for STI. This task is often performed by retrieving and scoring possible entity candidates (from a target KG) to disambiguate a cell value. Next, the result is used as input for performing Column-Type Annotation (CTA) and Columns-Property Annotation (CPA) [1,6,13,21]. However, associating a mention contained in a cell with an entity in a KG is a complex task requiring the resolution of several issues including handling properly the syntactic heterogeneity of mentions (e.g. the Wikidata entity "France" (Q142) may be referenced in a table by mentions like "The Republic of France" or "FRA"), the polysemy of terms (e.g. "Apple" can refer to a fruit or a company), and the diversity and complexity of table formats and layouts (e.g. matrices, relational table with hidden subjects, etc.).

Numerous approaches have been proposed for handling these issues. Among these methods, heuristic-based iterative approaches [1,6,13,21] aim to leverage the column types and the inter-column relationships aggregated by voting strategies for disambiguating cell annotations. They have demonstrated to be the methods reaching the best performance in the SemTab challenge series [16,17,9]. However, one drawback of these strategies is related to error propagation. Often, such systems propagate the entity annotation error in the target column. Furthermore, they also often ignore other column-wised semantic similarities: for example, books appearing in the same column may share the same topic.

To address these limitations, we propose a new hybrid disambiguation system called Radar Station that takes advantage of both an iterative disambiguation pipeline and semantic disambiguation using graph embedding similarities. Radar Station takes as input CEA annotations and associated confidence scores that quantify the level of certainty associated with each result. Our approach uses an ambiguity detection module that detects cases where the cell annotation is potentially wrong due to error propagation. In the following steps, the use of graph embeddings allows Radar Station to potentially fix the wrong annotations by taking into account semantic proximities (e.g. geometric proximity of entities representing books) that are not directly encoded and captured in the sole content of table columns. We evaluate Radar Station using several graph embedding models belonging to different families on Web tables as well as on synthetic datasets, and we provide a thorough analysis of the performance among the graph embeddings models and the datasets.

The remainder of this paper is structured as follows. In Section 2, we introduce some definitions and the assumptions we made in this study. Next, we review the many approaches that have been proposed for cell annotations and discuss their limitations (Section 3). In Section 4, we present the Radar Station system and the use of embeddings for cell-entity disambiguation. Then, we present our ex-

perimental settings in Section 5 and the evaluation using existing gold standards in Section 6. Finally, we conclude and outline future work in Section 7.

2 Preliminaries

In this paper, we focus on relational tables since they are the most used type of tabular data. Relational tables geometrically translate subjects from the same topic and their attributes accordingly to a given orientation. More specifically, in a horizontal table, a row describes the attributes of a given entity, and a column contains the values of a given attribute for all entities contained in the table. For example, Table 1 provides an example of relational table from the Limaye dataset [19]. The last row describes the book "Ylesia" with its attributes, including the published year ("2002") and the platform ("e-book"). We assume that

Table 1: Table file405599_0_cols1_rows23.csv from Limaye dataset, row 13-17

2002	Enemy Lines : Rebel Dream		
2002	Enemy Lines : Rebel Stand		
2002	Traitor		
2002	Destiny's Way		
2002	Ylesia		e - book

the orientation of the input table is known and is either horizontal or vertical. We also assume that a cell value does not contain more than a single entity and that the system knows the target columns containing the entities to annotate. Radar Station assumes that a table cell can always be correctly annotated with an entity w.r.t this KG. Given the above assumptions, Radar Station is a system that aims to improve cell disambiguation from annotations produced by an STI system using a target KG (Wikidata, in our experiments). Given Table 1, Radar Station annotates the cell "Traitor" with the entity "Q7833036", the science fiction book, using the table context, while often, traditional STI system will disambiguate this cell with the anti-war romance novel "Q21161161" which has the same label.

3 Related Work

STI covers five main tasks: CEA, CTA, CPA [16], row-to-instance annotation and table topic annotation [24]. Radar Station aims to improve the CEA disambiguation. Thus, this section reviews the current state-of-the-art methods for the CEA task on relational tables. We classify them into three groups: heuristic-based approaches, iterative disambiguation, and graph embeddings approaches, and we discuss their strengths and limitations [20].

3.1 Heuristic-Based Approaches

Starting from a basic lookup service that generates target candidates for a given cell mention, heuristic-based approaches leverage diverse methods interpreting the table context to filter unreliable candidates and to produce a final annotation. Based on heuristic candidate generation and string similarities measures, [19] is one of the first works on STI. It constructs a graph-based algorithm that exploits learnable features from column context, row context, and relation context to construct a confidence function for each candidate for annotating a cell. TabEL [3] introduces a hybrid system that leverages probabilities to build a graphical model for representing the interactions between cells, columns, and headers. ADOG [22] generates features from string similarities, frequencies of properties, and the normalized Elasticsearch score. Then, these features are calibrated with the candidate's TF-IDF score according to entities' types in the same column. Our approach takes as input a list of CEA candidate annotations together with their scores (generated by such an existing CEA annotation tool), and detects the presence of potential ambiguities in order to select the right candidate from this closed set.

3.2 Candidate Disambiguation

Adding a disambiguation process on top of a heuristic-based approach can significantly improve the performance of an annotation system. Iterative processing is one of the most commonly-used methods for improving pre-annotated results. The iteration loop aims to collect the results of several annotation tasks, mutually improving the compatibility between annotations (e.g. taking into account the type of a column produced by the CTA to choose the right CEA candidates), and increasing the scores of candidates that would not have been chosen in the first place. For example, [35] uses a loop that exploits the CTA annotation of a given column to select candidate cells that feature that type and then redefines a new CTA annotation for the column by exploiting the entities selected. Regarding the CEA disambiguation, we identified two classes of iterative systems. First, T2K [25] and TableMiner+ [33] introduce a loop in the pipeline that ends when the result becomes stable. The other iterative systems [1,6,13,21] provide a predefined pipeline with sequential modules (e.g. the pipeline of LinkingPark [6] is composed of a CEA pre-scoring, then a CPA step, and finally the use of the CPA annotations to generate the final CEA annotations).

Radar Station uses the output scores of an existing STI system. It currently supports the DAGOBAH-SL [14,13], MTab [21] and BBW [27] systems which have all competed during the SemTab Challenge series [17,9] and are selected as baseline systems during the evaluation of Radar Station. These systems use string similarity in the scoring system and leverage table's global information carried out by the CTA and CPA annotations to generate more precise CEA annotations. For cell annotations, they evaluate whether a candidate entity $e_c \in \mathcal{E}_c(e_m)$ retrieved from the KG is a good representation of the corresponding table cell e_m by incorporating the table context of e_m and KG context of e_c in the score of e_c.

Although these approaches show great performance for datasets like BioTable and HardTable from SemTab, they still have limitations as described in Section 1. First, the use of a unique column type or columns pairs relationship potentially propagates type (resp. relation) annotation error through cell annotations. Second, leveraging only entities' type (resp. relations) result does not allow to take into account more attributes and properties in the disambiguation process. For example, a column type may not bring necessary information such as a person's nationality, building localization, or object ownership for disambiguating entities. Facing these challenges, Radar Station first activates an ambiguity detection module that detects cases where the cell annotation is potentially wrong. Meanwhile, it considers entities' embeddings to leverage more similarity measures inside a given column.

3.3 Usage of Graph Embeddings

Methods applying graph embeddings for STI focus on entity-level in which the models learn embedding representations for entities of a table cell instead of the cell itself. Specifically, KG embedding techniques are used to encode the entities and their relationships into a vector space. STI approaches using deep learning models are based on the intuition that the entities in the same column should exhibit semantic similarities. Hence, they should be close to each other in the embedding space w.r.t. a cosine similarity distance [11] or an Euclidean distance [5].

Vasilis et al. [11] provide different methods. One of them assumes that the correct CEA candidates in a column should be semantically close. From this assumption, a weighted correlation subgraph in which a node represents a CEA candidate is constructed. The edges are weighted by the cosine similarity between two related nodes. The best candidates are the ones whose accumulated weights over all incoming and outcoming edges are the highest. In addition, a hybrid system combining a correlation subgraph method and an ontology matching system, is also introduced, which considerably improves the final result. Yasamin et al. [12] further enhance this approach by taking the header of the table into account for ontology matching and giving more weights to unique cell candidates when calculating embeddings Page-Rank. DAGOBAH-Embedding [5] follows the same assumption that all entities in the same column of the table should be close to each other in the embeddings space. Consequently, the correct candidates are assumed to belong to a few clusters. They apply a K-means clustering using TransE pre-trained KG embeddings to cluster the entity candidates. The good clusters with high coverage are selected by a weighted voting strategy. Experimental results prove that they have successfully improved the accuracy of the CTA task. However, the system is also misled by incorrect candidates during the CEA task when correct candidates are not in selected clusters. TURL [10] leverages the BERT model for STI and table augmentation with the help of a visibility matrix for capturing table structure. Although TURL introduces entity embeddings as one of the inputs to its model to assign information to entities,

the entity embeddings do not embed properties about the entities in the graph, such as the fact that neighbouring nodes are missing in them.

The contributions of our approach are as follows. First, we use embeddings only during the disambiguation step to benefit from both the iterative disambiguation and the embeddings disambiguation. Second, we provide a new scoring mechanism that takes into account the scores generated by CEA approaches and the distance between the entities in the embedding space.

4 System Description

Radar Station is not a standalone annotation system. It is built on top of a given annotation system and resolves ambiguities detected in the annotated results. We choose to use DAGOBAH-SL [26] as the base annotation system to illustrate the process of Radar Station. We motivate the need for Radar Station observing that pure string-based matching and iterative scoring methods are limited in situations where: i) the target KG is incomplete; ii) the matching mechanism failed; iii) the CTA or CPA disambiguation can not provide enough information in very ambiguous cases (e.g. candidates belonging to the same type or no property identified). These situations also cover cases with limited row numbers that can annotate a unique column type (resp. unique columns relationship) by majority voting. For example, voting for a common type given only the two cell mentions "Apple" and "Blackberry" may lead to randomly select the company or the fruit.

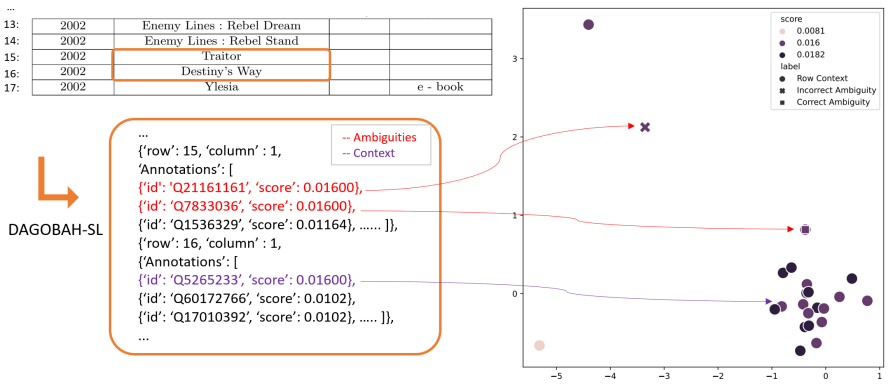


Fig. 1: Illustration of Radar Station with DAGOBAH-SL results. The plot is generated with RotatE embeddings after dimension reduction by T-SNE.

Figure 1 provides an example where DAGOBAH-SL is not able to handle properly an ambiguous case: two potential candidates with the same score for the cell "Traitor". The ambiguity comes from an unsuccessful matching between the

column “2002” with literal information “30 July 2002” of candidate “Q7833036” for entity scoring and CPA disambiguation. CTA disambiguation does not work in this case since these two candidates are books with the type “literary work” (“Q7725634”). “Q21161161” is a science fiction novel and “Q7833036”is an anti-wars romance novel.

We do not aim at improving the performance of the system by relying on clever string matching methods. Instead, we expect to find more semantic similarities using the full column as context with the help of the scores generated from the row context. In this example, one could identify that the correct entity is “Q7833036” since the topic of this table is the science fiction series “The New Jedi Order” from Star Wars. This relationship is missing in the table cells, but it still could be beneficial for the disambiguation steps. Radar Station aims to leverage graph embeddings to dig similarities alongside the entity types and common relationships inside the tables. The architecture of Radar Station is illustrated in Figure 2 and the modules are described in the following sections. Table 2 summarizes the notation used in the Radar Station approach.

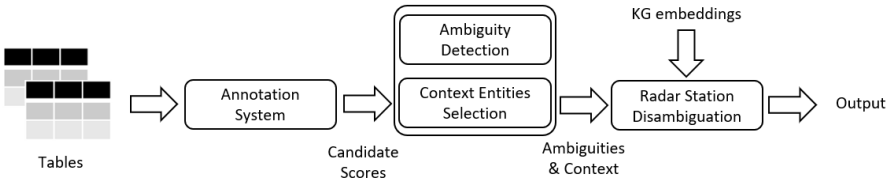


Fig. 2: Overview of the Radar Station pipeline.

Table 2: Summary of the notation used to define Radar Station

Notation	Description
$\mathcal{C}$	The collection of cells from the target column
$\mathcal{E}_c$	The collection of the context entities representing the column $\mathcal{C}$
$\mathcal{A}m_{c_i}$	The collection of the ambiguous entities extracted from the cell c_i
$\mathcal{S}c(e)$	The initial score for the candidate e generated from the previous annotation system, in our case, DAGOBAH-SL
$\mathcal{E}m(e)$	The embedding of a given entity e
$\mathcal{E}_k$	The collection of K nearest context entities of an ambiguous candidate $am \in \mathcal{A}m_{c_i}$ for the target column $\mathcal{C}$, $\mathcal{E}_k \in \mathcal{E}_c$

4.1 Input Data Structure

Before running Radar Station, the information required by the system includes the index of the cell in the table (row number and column number), and information about all candidates for each cell without filtering the candidates. This

information includes an identification of each candidate and their confidence score. The confidence score evaluates how compatible a candidate is with the context information given by the table (e.g. row values, column type, columns-pair relations).

4.2 Context Entities Selection

The row context has already been interpreted by DAGOBAH-SL and is used to compute the confidence score of each candidate. The first step of Radar Station is to build a column-wised context to support the disambiguation process. We collect entities with the highest confidence score from all cells of a given column $\mathcal{C}$ as the context entities set. In case of ambiguity, that is, multiple candidates (n candidates) sharing the same highest score, we collect all of them, and the score is divided by n. Other candidates are not taken into account to maximize the trust for "sure" annotation from DAGOBAH-SL (e.g., only one candidate with the highest score) and to avoid noise inside this column. For example, for the row 15 in Figure 1, both "Q7833036" and "Q21161161" are collected into the context set with a score "0.008" (0.016/2), and for row 16, only "Q5265233" is collected with a score "0.016". The collected context entities set for the column $\mathcal{C}$ are noted as $\mathcal{E}_c$.

4.3 Ambiguity Detection

Radar Station detects ambiguous cells that are worthy to be disambiguated given a tolerance t. Intuitively, t enables to relax the constraints one wants to have in looking up candidates potentially matching a cell mention. Once a candidate's score is larger than $t * Max(scores)$, it is selected as one of the "top candidates". For example, if we set $t = 1$, "Q7833036" and "Q21161161" for row 15 of Figure 1 will be among the top candidates. If we relax the tolerance t to 0.7, "Q1536329" will also be considered as a top candidate. We denote "Ambiguities" as Am for the case that the size of the top candidates is greater than or equal to two. Radar Station is activated in this case and it will annotate the cell with one of the candidates from the ambiguities. When there is no ambiguity inside a cell, we directly output the single top candidate as the annotation.

4.4 Radar Station Disambiguation

In our approach, we leverage KG embeddings to uncover the entities' co-relationship from a table to improve the disambiguation step. The principle of the Radar Station approach is inspired by radar station signal emissions. The receiving signal power of a signal station depends on both the initial power strength from the sending station and the distance between the sender and the receiver. That is, the receiving signal will be stronger when the initial power from the sender is stronger, and this receiving signal strength will decrease as the distance increases. In our approach, we treat each context entity from the same column as a signal

Algorithm 1 Radar Station disambiguation algorithm

Input: Cell index $\mathcal{C}$ and ambiguities for each cell $\mathcal{A}m_{c_i}$, $c_i \in \mathcal{C}$ where the collected context entities (or senders) of the target column is $\mathcal{E}_c$.
Candidate scores from the annotation system $\{Sc(e_i)\}$, $e_i \in \mathcal{E}_c$.
Candidate embeddings $\{Em(e_i)\}$, $e_i \in \mathcal{E}_c$.
Output: Entity annotation selected by Radar Station.

1: build a KD-tree with all candidates' embeddings $\{Em(e_i)\}$
2: $K \leftarrow min(|\mathcal{E}_f|, 20)$
3: **for** each cell c_i from $\mathcal{C}$ **do**
4: **if** there is an ambiguity in $\mathcal{A}m_{c_i}$ **then**
5: $\mathcal{E}_c \leftarrow$ filter entities from the same cell in $\mathcal{E}_c$
6: **for** each ambiguous entity am_i in $\mathcal{A}m_{c_i}$ **do**
7: find the K nearest candidates of the ambiguous entity $\mathcal{E}_k$ in the KD-tree by ignoring candidates from the same cell.
8: $RadarScore_{am_i} \leftarrow 0$
9: **for** each neighboring entity $e_j \in \mathcal{E}_k$ **do**
10: $RadarScore_{am_i} \leftarrow RadarScore_{am_i} + \frac{Sc(e_j)}{distance(am_i, e_j)}$
11: **end for**
12: $RadarScore_{am_i} \leftarrow \frac{RadarScore_{am_i}}{K}$
13: $g(am_i) \leftarrow \alpha RadarScore_{am_i} + Sc(e_j)$
14: **end for**
15: the annotation is the ambiguous entity with the highest $g(am_i)$
16: **end if**
17: **end for**

sender, and receivers are the ambiguities to be resolved. One ambiguous candidate captures signals from multiple neighbouring context entities (i.e. senders) and the sum of the receiving signals is the confidence score of the candidate. The disambiguation pseudo-code is presented in Algorithm 1.

We consider only the K nearest context entities to computer the final score of an annotation in order to avoid noise and to optimize the performance. The system first constructs a KD tree of all context entities for each column, and then calls this KD tree to drop the K nearest context entities during the prediction (lines 1-2). We set that the maximum K value is 20 (line 2). We set the initial sender power strength with the confidence score generated by DAGOBAH-SL. One ambiguous candidate am_i detects K received signals from the surrounding senders $e_j \in \mathcal{E}_c$ (or context candidates) to generate the confidence score $f(am_i)$ with the Function 1 (line 3-12), where $Sc(e_j)$ denotes DAGOBAH-SL scores of the sender e_j and $distance(am_i, e_j)$ denotes the Euclidean distance between the sender e_j and the receiver am_i.

In detail, for a target am_i, we collect its top-K nearest neighbors from $\mathcal{E}_c$, where each context entity c_j belongs to $\mathcal{E}_c$. We have each context entity's scores $Sc(c_j)$ and the distance with the target candidate $distance(am_i, c_j)$. We then apply the Function 1. Like this, we could generate a confidence score for each of those two target candidates. We divide each of their context entities' confidence score

by the distance between those two candidates and then calculate the sum to compare.

$$f(am_i) = \frac{1}{K} \sum_{j<K} \left(\frac{Sc(e_j)}{distance(am_i, e_j)}\right) \quad (1)$$

The final result $g(am_i)$ for an ambiguous entity is the combination of Radar Station score $f(am_i)$ and the initial DAGOBAH-SL confidence score $Sc(am_i)$ introduced in Function 2 (line 13).

$$g(am_i) = \alpha f(am_i) + Sc(am_i) \quad (2)$$

Our initial experiments showed that the average distance in the embedding space between the target ambiguity and its top K nodes is approximately 1. According to the Function 1, we know that $f(am_i)$ and $S_c(am_i)$ are roughly in the same order of magnitude. Since we expect to disambiguate candidates with a tolerance between 0.7 and 1, we need the value of the discrepancy caused by $f(am_i)$ to be roughly within $S_c(am_i) * 0.3$. We originally set α to 0.3 and we tested the following α values (0,3, 0.2, 0.1, 0.05, and 0.01). We empirically observed that 0.05 gives the best results.

5 Experiments

In our experiments, we consider Wikidata as the target KG. We first rely on the DAGOBAH-SL system to lookup for candidates for each entity cell. We only consider the top 100 candidates according to the string similarity on entity label and aliases. We evaluate the result on four different gold standard datasets: T2D [25], Limaye [19], Tough Tables version 2 [8] and ShortTables.

5.1 Knowledge Graph Embeddings

Pre-trained KG embeddings can provide additional information for table understanding beyond the table context. Entities inside the same table column should be somehow co-related, which means they may share the same entity type, similar topics, or even attributes. In order to have the most suitable embeddings given the latest version of Wikidata, we use the PyTorch-BigGraph framework [18] for training embeddings. The triples used for the training are collected from a Wikidata dump published in May 2021[3]. Before the training, the triples with literal values and Wikimedia disambiguation page entities (e.g. "Q1151870") are filtered out. The selection of the final embeddings is made given our empirical evaluation of Radar Station after fine-tuning the hyper-parameters. We consider two representative translational distance models (TransE [4] and RotatE [28]) and two semantic matching models (DistMult [31] and ComplEx [29]) following the classification of [30].

[3] https://archive.org/details/wikibase-wikidatawiki-20210521

Translational distance models study the geometric distance between entities inside the vector space. TransE [4] considers both entities and relations from the same vector space. The training intends to adjust the three vectors from a given triple (h, r, t) to the synchronized state until $h + r \approx t$. In Pytorch-BigGraph, we use the `translation` operator for generating the TransE model. Unlike TransE's translation, RotatE [28] regards the relation as a rotational degree between heads and tails. It introduces a loss function based on $h \circ r \approx t$ for simulating the relation translation. We use the GraphVite's [34] pre-trained RotatE embeddings in our experiments.

Semantic matching models measure the similarity between entities and relations during the training. DistMult [31] is based on a bilinear scoring function $h^T M_r t$, where M_r is the relation matrix built on top of the entity. ComplEx [29] can be seen as a constrained variant of RESCAL [15] that leverages fewer relation dimensions inside a complex space. The ComplEx score is defined as $Re(h^T diag(r)\bar{t})$. In Pytorch-Biggraph training, we use the `diagonal` operator for generating DistMult embeddings and iterations between `complex_diagonal` and `dot` operators for ComplEx embeddings.

5.2 Datasets

We evaluate Radar Station on three popular gold standards: T2D[4], Limaye[5], and Tough Tables version 2[6]. The original T2D and Limaye datasets contain some annotation errors that we have corrected. As T2D and Limaye are gold standards based on DBpedia and Radar Station is a Wikidata-based annotation system, we translate the DBPedia entities given in the gold standards into Wikidata entities through the "Wikidata item" hyperlink from Wikipedia pages of DBpedia entities. We manually corrected this translation when it was failing. Since the number of entities in Wikidata is larger than the number of entities in DBpedia [23], the annotation based on Wikidata is also harder with more candidates to disambiguate. We publish the new resulting ground truth on Zenodo (see the supplementary material). ShortTables is a new dataset we built from T2D, in such a manner that each table only contains two rows. The aim of creating such a dataset is to simulate extreme cases where voting strategies lack electors (i.e. row entities) for a correct CTA (resp. CPA) annotation. The provenance of T2D and Limaye is Web tables. We also consider a synthetic dataset named Tough Tables version 2 (2T_2) to evaluate on more data types. We provide the statistics of these gold standard datasets in Table 3.

6 Evaluation

We evaluate Radar Station with these four datasets varying the embeddings and the tolerance threshold. A random selection of the highest scoring candidates

[4] http://webdatacommons.org/webtables/goldstandardV2.html
[5] http://websail-fe.cs.northwestern.edu/TabEL/
[6] https://zenodo.org/record/6211551

Table 3: Gold standard datasets for evaluating STI approaches. The ambiguities are based on DAGOBAH-SL scores

Gold standard	#Tables	Avg. #Rows	Avg. #Col	#Entities	Ambiguities (t=1)	Ambiguities (t=0.9)
Limaye	437	37	2	5,143	181 (3.52%)	685 (13.31%)
T2D	762	157	5	18,589	2,322 (12.49%)	8,852 (47.62%)
2T_v2	180	1080	5	661,297	30,686(4.64%)	86,739(13.11%)
ShortTables	2237	2	5	4,474	1422 (31.78%)	1822 (40.72%)

is considered as our baseline and noted as the original system name. We show the overall result for t equals to 1, 0.95, and 0.9 based on DAGOBAH-SL scores on four datasets with different embeddings in Table 4 and the fine-tuned result based on DAGOBAH-SL, MTab and BBW with Limaye and T2D in Table 5.

6.1 Evaluation Settings

We aim to evaluate the performance of Radar Station on the ambiguity lists and how it can influence the global annotations. Thus, we use three indicators including Ambiguity quality (AP), Precision inside ambiguities (PA), and Global precision (GP). AP (Equation 3) shows the quality of generated ambiguity list after the Ambiguity Detection step, that is, how many ambiguous cells contain a ground truth in its top candidates. It indicates the extreme precision that we could achieve in all ambiguous annotations, which is PA in Equation 4. GP (Equation 5) is the overall precision in all labelled cells considering annotations generated with or without Radar Station.

$$AP = \frac{\#Correct\ candidates\ in\ the\ candidate\ set\ of\ ambiguities}{\#\ Ambiguities} \tag{3}$$

$$PA = \frac{\#\ Correct\ ambiguity\ disambiguations}{\#\ Ambiguities} \tag{4}$$

$$GP = \frac{\#Correct\ annotations}{\#Total\ labels} \tag{5}$$

We also use the Cohen's Kappa coefficient [7] to evaluate the independence of the annotation from different embeddings models (*kappa* equals to 1 means that two datasets are the same).

6.2 Analysis

Overall result. We first observe from Table 4 that all the chosen embeddings contribute to a significant improvement for PA in the ambiguous cases with the chosen tolerance values and GP. We also notice that Radar Station brings more improvements to GP for the Limaye (Max. 0.02), T2D (Max. 0.03), and

Table 4: Radar Station evaluation based on DAGOBAH-SL scores. AP: Ambiguity quality, PA: Precision inside ambiguities, GP, Global precision

t	Methods	Limaye			T2D			2T_v2			ShortTables		
		AP	PA	GP	AP	PA	GP	AP	PA	GP	AP	PA	GP
1	DAGOBAH-SL	0.647	0.168	0.853	0.308	0.053	0.785	0.067	0.023	0.870	0.672	0.194	0.654
	RS + TransE		0.630	0.870		0.294	0.813		0.041	0.871		0.355	0.673
	RS + RotatE		0.636	0.870		0.289	0.812		0.044	0.871		0.363	0.673
	RS + DistMult		0.391	0.861		0.163	0.798		0.034	0.870		0.229	0.658
	RS + ComplEx		0.57	0.869		0.171	0.798		0.036	0.870		0.235	0.659
0.95	DAGOBAH-SL	0.614	0.296	0.853	0.332	0.180	0.785	0.327	0.208	0.870	0.671	0.302	0.654
	RS + TransE		0.528	0.872		0.312	0.815		0.230	0.872		0.414	0.673
	RS + RotatE		0.542	0.873		0.312	0.815		0.235	0.872		0.418	0.674
	RS + DistMult		0.377	0.860		0.230	0.797		0.213	0.870		0.328	0.659
	RS + ComplEx		0.435	0.864		0.233	0.798		0.219	0.870		0.334	0.660
0.9	DAGOBAH-SL	0.653	0.432	0.853	0.336	0.241	0.785	0.500	0.300	0.870	0.714	0.414	0.654
	RS + TransE		0.570	0.872		0.323	0.815		0.313	0.872		0.532	0.684
	RS + RotatE		0.578	0.873		0.322	0.814		0.318	0.872		0.536	0.684
	RS + DistMult		0.475	0.860		0.274	0.797		0.303	0.870		0.466	0.668
	RS + ComplEx		0.494	0.862		0.275	0.798		0.306	0.870		0.471	0.669

Table 5: Gold standard datasets for evaluating STI approaches with RotatE embeddings. AP: Ambiguity quality, PA: Precision inside ambiguities, GP, Global precision

Dataset	System	t	AP	Original output		Radar Station	
				PA	GP	PA	GP
Limaye	DAGOBAH-SL	0.9	0.653	0.432	0.853	0.578 (+0.146)	0.873 (+0.020)
	MTab	0.83	0.820	0.705	0.857	0.787 (+0.082)	0.875 (+0.018)
	BBW	0.65	0.587	0.359	0.563	0.507 (+0.148)	0.597 (+0.034)
T2D	DAGOBAH-SL	0.95	0.332	0.180	0.785	0.312 (+0.132)	0.815 (+0.030)
	MTab	0.71	0.385	0.295	0.837	0.346 (+0.051)	0.857 (+0.020)
	BBW	0.65	0.263	0.192	0.364	0.253 (+0.061)	0.382 (+0.018)

ShortTables (Max. 0.03) than for 2T_v2 (Max. 0.002). This drop for 2T_v2 is due to the distribution of the scores of the top candidates: i) as we can see, after relaxing the tolerance from 1 to 0.9, AP for 2T_v2 has dramatically increased in comparison to the other datasets. Hence, there is no clear boundary between top candidates and bad candidates for the 2T_v2 dataset. That leads to a relatively lousy context embedding for the disambiguation. This scoring distribution is impacted by row number with DAGOBAH-SL mechanism, that is, the more rows we have, the more balanced the scoring would be; ii) the other reason is that 2T_v2 is a synthetic dataset generated with types from a KG. Thus, other column-wised semantic similarities are not obvious in this dataset. Hence, we recommend that future synthetic datasets should consider the inclusion of common themes from these tables to simulate other real-world use cases.

We introduce ShortTables for simulating the extreme cases where the very limited number of rows does not allow existing systems to generate correct CTA

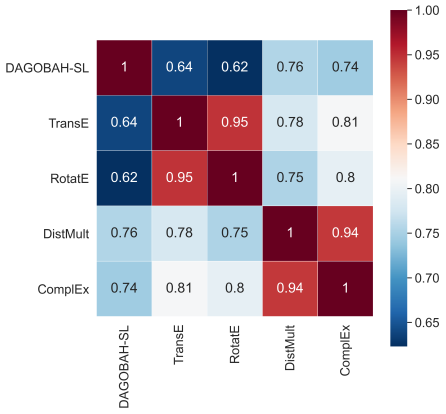
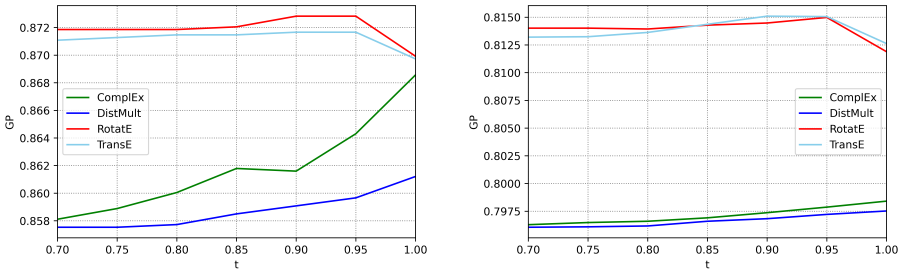


Fig. 3: Illustration of the Kappa test between different outputs on all datasets, $t = 0.95$.

and CPA annotations. Bad CTA or CPA may propagate the error to the cell annotations. Thus, we expected to have a more significant GP improvement for ShortTables compared to T2D. However, from our evaluation, the contribution of Radar Station is close in these two datasets (Max 0.03). We analyze that a small number of rows can decrease the quality of type annotation and more likely propagate error with type disambiguation: therefore, it provides more chances for semantic disambiguation. At the same time, the limited number of rows also limits the content of the context entity set that has been used for semantic disambiguation. We argue that these two effects cancel each other in this experiment. We have implemented Radar Station on two other systems and evaluate its performance with two Web table datasets. The result shown in Table 5 indicates that Radar Station benefits to all input annotation systems.

Analysis on embeddings. Regarding the two families of embeddings (TransE and RotatE are translational distance models, DistMult and ComplEx are semantic matching models), the GP for embeddings from the same family achieves similar results inside our trained embeddings. From the result of Cohen's kappa shown in Figure 3, we observe that the output is similar for embeddings from the same family. For example, the kappa value for TransE and RotatE is much higher than TransE with other outputs (same for DistMult and ComplEx). This similarity could also be seen in the precision shown in Table 4. We also observe that translational distance models are generally better than semantic matching models in our trained embeddings. That may be because we leverage geometric distance inside Radar Station, which is compatible with the training strategy of translational distance models. Globally, RotatE embeddings outperform all other models for all datasets.

Tolerance. Relaxing the tolerance has for effect to include more candidate entities and thus has the potential to increase the probability that the correct candidate is in the candidate set. However, such an operation also puts more noise into the candidate list. In Figure 4, we illustrate how the tolerance influences the performance of the system on Limaye and T2D. It shows that relaxing the tolerance with TransE and RotatE improves the quality of the annotation (performance peak at t=0.95 in Figure 4). In our observation, largely relaxing the tolerance may decrease the accuracy since more noise is included during the disambiguation. This is therefore a delicate tradeoff to generalize across datasets.



(a) Limaye dataset (b) T2D dataset

Fig. 4: The GP evaluation on Limaye and T2D with t from 0.7 to 1 based on DAGOBAH-SL.

7 Conclusion and Future Work

In this paper, we analyze the current limitations of STI systems with relational tables and we introduce Radar Station, a new disambiguation method that makes use of pre-trained KG embeddings to strengthen the performance. We evaluate the system with different embeddings methods and we prove that this optimization can be beneficial. In the future, we aim to process more table types such as entity tables or matrix tables for enhancing the coverage and robustness of the system. We also seek to leverage language models to take more contextual information into account. We finally aim to plug Radar Station on top of other competitive STI systems.

Supplemental Material Statement. The source code for Radar Station is available at https://github.com/Orange-OpenSource/radar-station. The RotatE Embeddings, TransE embeddings, DAGOBAH-SL scores, Ground Truth and other required datasets are available from Zenodo at https://zenodo.org/record/6522985 while the ComplEx and DistMult embeddings are available at https://zenodo.org/record/6522921.

References

1. Abdelmageed, N., Schindler, S.: JenTab: Matching Tabular Data to Knowledge Graphs. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). pp. 40–49 (2020)
2. Bhagavatula, C.S., Noraset, T., Downey, D.: Methods for exploring and mining tables on wikipedia. In: ACM SIGKDD Workshop on Interactive Data Exploration and Analytics. pp. 18–26 (2013)
3. Bhagavatula, C.S., Noraset, T., Downey, D.: Tabel: Entity linking in web tables. In: 14^{th} International Semantic Web Conference. pp. 425–441. Springer (2015)
4. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: International Conference on Advances in Neural Information Processing Systems (NIPS). vol. 26 (2013)
5. Chabot, Y., Labbe, T., Liu, J., Troncy, R.: DAGOBAH: an end-to-end context-free tabular data semantic annotation system. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). pp. 41–48 (2019)
6. Chen, S., Karaoglu, A., Negreanu, C., Ma, T., Yao, J.G., Williams, J., Gordon, A., Lin, C.Y.: Linkingpark: An integrated approach for semantic table interpretation. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) (2020)
7. Cohen, J.: Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. Psychological bulletin **70**(4), 213 (1968)
8. Cutrona, V., Bianchi, F., Jiménez-Ruiz, E., Palmonari, M.: Tough tables: Carefully evaluating entity linking for tabular data. In: International Semantic Web Conference (ISWC). pp. 328–343. Springer (2020)
9. Cutrona, V., Chen, J., Efthymiou, V., Hassanzadeh, O., Jiménez-Ruiz, E., Sequeda, J., Srinivas, K., Abdelmageed, N., Hulsebos, M., Oliveira, D., et al.: Results of SemTab 2021. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). pp. 1–12. CEUR Workshop Proceedings (2022)
10. Deng, X., Sun, H., Lees, A., Wu, Y., Yu, C.: TURL: Table Understanding through Representation Learning. arXiv:2006.14806 (2020)
11. Efthymiou, V., Hassanzadeh, O., Rodriguez-Muro, M., Christophides, V.: Matching web tables with knowledge base entities: from entity lookups to entity embeddings. In: 16^{th} International Semantic Web Conference (ISWC). pp. 260–277. Springer (2017)
12. Eslahi, Y., Bhardwaj, A., Rosso, P., Stockinger, K., Cudré-Mauroux, P.: Annotating web tables through knowledge bases: A context-based approach. In: 7^{th} Swiss Conference on Data Science (SDS). pp. 29–34. IEEE (2020)
13. Huynh, V.P., Liu, J., Chabot, Y., Deuzé, F., Labbé, T., Monnin, P., Troncy, R.: DAGOBAH: Table and Graph Contexts for Efficient Semantic Annotation of Tabular Data. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) (2021)
14. 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) (2020)
15. Jenatton, R., Le Roux, N., Bordes, A., Obozinski, G.: A latent factor model for highly multi-relational data. In: International Conference on Advances in Neural Information Processing Systems (NIPS). pp. 3176–3184 (2012)

16. Jiménez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., Srinivas, K.: SemTab 2019: Resources to Benchmark Tabular Data to Knowledge Graph Matching Systems. In: European Semantic Web Conference (ESWC). pp. 514–530. Springer (2020)
17. Jiménez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., Srinivas, K., Cutrona, V.: Results of SemTab 2020. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). vol. 2775, pp. 1–8 (2020)
18. Lerer, A., Wu, L., Shen, J., Lacroix, T., Wehrstedt, L., Bose, A., Peysakhovich, A.: Pytorch-biggraph: A large scale graph embedding system. In: Conference on Machine Learning and Systems (MLSys). vol. 1, pp. 120–131 (2019)
19. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. Proceedings of the VLDB Endowment **3**(1-2), 1338–1347 (2010)
20. Liu, J., Chabot, Y., Troncy, R., Huynh, V.P., Labbé, T., Monnin, P.: From Tabular Data to Knowledge Graphs: A Survey of Semantic Table Interpretation Tasks and Methods. Journal of Web Semantics (2022), under revision
21. 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) (2020)
22. Oliveira, D., d'Aquin, M.: Adog-annotating data with ontologies and graphs. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) (2019)
23. Ringler, D., Paulheim, H.: One knowledge graph to rule them all? Analyzing the differences between DBpedia, YAGO, Wikidata & co. In: Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz). pp. 366–372. Springer (2017)
24. Ritze, D., Bizer, C.: Matching Web Tables To DBpedia - A Feature Utility Study. In: International Conference on Extending Database Technology (EDBT). pp. 210–221 (2017)
25. Ritze, D., Lehmberg, O., Bizer, C.: Matching html tables to dbpedia. In: 5^{th} International Conference on Web Intelligence, Mining and Semantics. pp. 1–6 (2015)
26. Sarthou-Camy, C., Jourdain, G., Chabot, Y., Monnin, P., Deuzé, Huynh, V.P., Liu, J., Labbé, T., Troncy, R.: DAGOBAH UI: A New Hope For Semantic Table Interpretation. In: 19^{th} European Semantic Web Conference (ESWC), Poster and Demo Track. Springer (2022)
27. Shigapov, R., Zumstein, P., Kamlah, J., Oberländer, L., Mechnich, J., Schumm, I.: bbw: Matching CSV to Wikidata via Meta-lookup. In: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) (2020)
28. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. arXiv:1902.10197 (2019)
29. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International Conference on Machine Learning (ICML). pp. 2071–2080. PMLR (2016)
30. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering **29**(12), 2724–2743 (2017)
31. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv:1412.6575 (2014)
32. Zhang, S., Balog, K.: Recommending related tables. arXiv:1907.03595 (2019)

33. Zhang, Z.: Effective and efficient semantic table interpretation using tableminer+. Semantic Web **8**(6), 921–957 (2017)
34. Zhu, Z., Xu, S., Tang, J., Qu, M.: Graphvite: A high-performance cpu-gpu hybrid system for node embedding. In: The World Wide Web Conference (WWW). pp. 2494–2504 (2019)
35. Zwicklbauer, S., Einsiedler, C., Granitzer, M., Seifert, C.: Towards Disambiguating Web Tables. In: International Semantic Web Conference (ISWC), Posters & Demos Track. pp. 205–208 (2013)