DAGOBAH: Enhanced Scoring Algorithms for Scalable Annotations of Tabular Data

Viet-Phi Huynh[1], Jixiong Liu[1], Yoan Chabot[1], Thomas Labbé[1], Pierre Monnin[1], and Raphaël Troncy[2]

[1] Orange Labs, France
yoan.chabot@orange.com
[2] EURECOM, Sophia Antipolis, France
raphael.troncy@eurecom.fr

Abstract. We present new approaches used in the DAGOBAH system to perform automatic semantic table interpretation. DAGOBAH semantically annotates tables with Wikidata entities and relations to perform three tasks: Columns-Property Annotation (CPA), Cell-Entity Annotation (CEA) and Column-Type Annotation (CTA). In our system, the initial scores from entity disambiguation influence the CPA output, which, in turn, influences the output of the CEA. Finally, the CTA is computed using the type hierarchy available in the knowledge graph in order to annotate columns with the most suitable fine-grained types. This approach that leverages mutual influences between annotations allows DAGOBAH to obtain very competitive results on all tasks of the SemTab2020 challenge.

Keywords: Tabular Data · DAGOBAH · SemTab Challenge

1 Introduction

Within the ever-expanding Web of data, more and more knowledge graphs (KGs) become available. However, these KGs may suffer from inconsistency and incompleteness issues [20]. Hence, one can envision to either correct or complete KGs by extracting information from various sources such as web tables and texts available in Web pages [7]. Interestingly, tables often constitute a major source of information since large parts of both companies internal repositories and Web pages are represented in tabular formats. Additionally, besides KG completion, the automatic interpretation of tables by software agents can enable semantic-driven services to query, manipulate, and process heterogeneous table corpora [2], such as a dataset search "moving beyond keyword" [4].

These automatic annotation tasks raise several challenges. For example, tables present limited context to resolve semantic ambiguities, and their layout can be complex (*e.g.* merged rows, missing headers, table orientation to determine). The aforementioned perspectives and challenges have motivated the development of numerous approaches that perform table annotation with entities and relations from a KG via three main tasks: Cell-Entity Annotation (CEA), Column-Type Annotation (CTA) and Columns-Property Annotation (CPA) [12]. The previous SemTab2019 challenge edition has structured this effort, enabling systems to be compared on common datasets and metrics. We observed that systems based on optimized lookups and majority voting were largely represented [5, 16, 19, 23, 24]. Other works attempt different approaches using probabilistic models [18] or disambiguation with embeddings [3].

During the SemTab2019 challenge, we first developed and evaluated the DAGOBAH system [3], a generic annotation system that can handle large corpora of heterogeneous tables. In the context of the SemTab2020 challenge, we improved DAGOBAH with the following main contributions:

– Mutual influences between the CEA and CPA tasks to resolve ambiguities;
– A fine-grained typing of columns (CTA) by considering the KG type hierarchy;
– An attention on the scalability of the system for an efficient execution on large table corpora;
– A continuous exploration of the role of knowledge graph embeddings for the CEA and CTA tasks.

The remainder of this paper is organized as follows. In Section 2, we introduce some related approaches tackling the tasks associated with tabular data annotation. In Section 3, we detail the algorithms used in the DAGOBAH system, whose performance on the SemTab2020 corpus is presented and discussed in Section 4. Finally, we outline some future works in Section 5.

2 Related Work

Two main approaches are used to carry out the CEA task. The most common approach consists in finding a match for each cell of a given table using syntactic lookups (*e.g.* Levenshtein distance), aligning ontologies, or exploiting embeddings [1, 9, 13, 15]. The second approach, on the other hand, attempts to match an entire row of a table with an entity of the target KG based on the assumption that a row represents a main entity (key column) and associated attributes (other columns) [8]. The disambiguation of the candidate entities is then treated as a typical ranking task among the candidates, using either heuristics or algorithms such as PageRank [8], similarity measurement [9, 13], or graph-based models [11]. The main approaches on column typing (CTA) infer classes from the entities output by the CEA task. Different algorithms have been proposed, integrating more or less complex heuristics built around majority voting [17]. Finally, the extraction of relations (CPA) is generally carried out by searching pairs of elements in columns, *i.e.* types and entities previously determined [21].

If the previous approaches are intrinsically sequential, we observe that some approaches aim to jointly achieve the three tasks via graph-oriented models [15, 25].

In parallel, efforts have been made by the community to provide evaluation corpora for these different tasks. While the first corpus only contained 428 Wikipedia tables [15], the following ones were expanded either in volume (WTC [14]: 233 million tables divided into three categories) or in precision (T2D Gold Standard [22]: 1748 tables from the WTC whose rows, attributes, and tables were manually annotated with instances, properties, and DBpedia classes respectively). Some corpora have also been customized to meet a particular need (Wikipedia Gold Standard [8]) but do not currently serve as an evaluation reference within the community.

More recently, the SemTab2019 challenge invited research teams to compare the performance of their tabular data annotation systems for the three tasks mentioned above. The challenge took place in four rounds [12] proposing corpora differing in size (respectively 63, 11925, 2162, and 818 tables), and nature (increasingly complex tables both in terms of formats and information to be processed) [10]. Seventeen teams participated in the challenge with seven of them having participated in at least two of the four rounds of the challenge, including the DAGOBAH system. A majority of the approaches take the form of systems based on the search of candidates in DBPedia and Wikidata, the calculation of a syntactic similarity and majority votes [5, 16, 19, 23, 24]. IDLab proposes an iterative approach in which the annotation of cells with little ambiguity reinforces the disambiguation of more complex cells over the iterations [23]. Tabularisi uses the aliases of the entities defined in Wikidata and ensures that the columns have a semantic consistency at the end of the annotation process [24]. ADOG creates a DBpedia index in ArrangoDB and essentially uses the Levenstein distance to measure the difference between cell values and labels of entities [19]. MantisTable differentiates itself with a sophisticated graphical interface to configure the automatic annotation process and to view the results [5]. LOD4ALL essentially uses SPARQL ASK queries to obtain candidates and deduce type constraints on the columns [16]. Alternatively, MTab [18] proposes the use of a probabilistic approach coupled with queries on several search services on DBpedia, Wikidata, and Wikipedia using multilingual strategies. Evaluations show that approaches using sophisticated search techniques and optimized cleaning and pre-treatment steps achieve better results.

3 DAGOBAH: An Enhanced and Scalable Table Annotation System

The DAGOBAH system is designed to interpret relational tables by annotating them with entities and relations from the Wikidata knowledge graph (KG). It mostly consists of the following steps (Fig. 1). Given a relational table as input, the entity lookup module retrieves candidate entities from the KG for each target cell (Section 3.1). The pre-scoring module evaluates each candidate with a confidence score (Section 3.2). Then, the Columns-Property Annotation (CPA) is carried out (Section 3.3). Its output is used during the Cell-Entity Annotation (CEA) to disambiguate entities (Section 3.4). Finally, the Column Type Annotation (CTA) uses the output entities and a majority vote to annotate columns with types (Section 3.5).

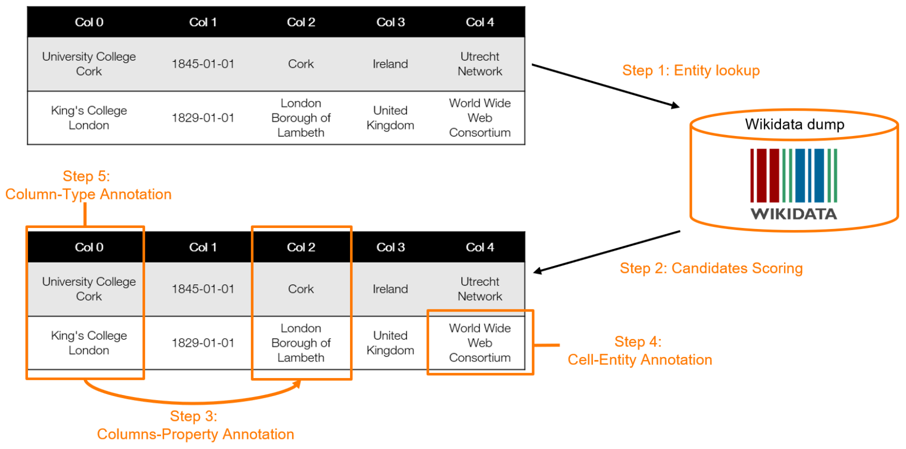

Col 0
Col 1
Col 2
Col 3
Col 4
University College Cork
1845-01-01
Cork
Ireland
Utrecht Network
King's College London
1829-01-01
London Borough of Lambeth
United Kingdom
World Wide Web Consortium
Step 1: Entity lookup
Wikidata dump
WIKIDATA
Step 2: Candidates Scoring
Step 3: Columns-Property Annotation
Step 4: Cell-Entity Annotation
Step 5: Column-Type Annotation

Fig. 1. Overview of the DAGOBAH annotation workflow. The table “3BHPG0ZP.csv” comes from the Round 3 of the SemTab2020 challenge.

An embedding approach has also been implemented, based on the intuition that semantically similar entities can be clustered in a vector space, allowing geometrical disambiguation. The implementation is similar to the one described in [3] but was only used in Round 1. This approach is later referenced as “DAGOBAH Embedding”. In the details, we use the K-Mean algorithm to cluster the candidate lookups for a column into N partitions. N is defined as the ratio between the total number of lookups and the number of table rows. We then select the three best clusters based on their coverage and we score each entity candidate belonging to those clusters. The entity scoring function takes the cluster confidence into account, apart from the confidence score of the entity itself. However, we observed that its added-value was limited since tables provided by the challenge were hardly noisy (as discussed in Section 4). In the remainder of this paper, we describe our so-called “DAGOBAH_SL(Semantic Lookup)” method which contains an algorithmic optimization of the scoring function.

3.1 Entities Lookup

Given a target cell[3] e_m contained in a table, we aim at retrieving a set of relevant candidates $\mathcal{E}_c$ from a KG, *i.e.* entities whose labels or aliases are similar to the text of the cell. We employ two strategies to evaluate the similarity between a cell and the candidate entities: an exact match using a regex similarity and a threshold-based match using the Levenstein ratio. We take the maximum value between all comparisons made across entity labels and aliases.

Regex similarity Candidate labels or aliases should include all the words of e_m in any order. This is particularly useful to match the full name of a person since the first and last names could appear in any order, *e.g.* "Elon Musk" versus "Musk Elon". A full edit ratio on these two multiple tokens string would yield the relatively low score of 0.44.

Levenshtein ratio This ratio is computed between the candidate labels and aliases, and the text of the cell. We empirically fix the threshold to 0.65 to ensure that a cell has at least one candidate and we then retain the top 50 candidates for each cell.

Table 1. Overview of the SemTab2020 table corpus in each round. Several cells may contain the same text. Such cells are only counted once in the line "# Unique cells to annotate". The length of cells counts the number of characters.

	Round 1	Round 2	Round 3
# Tables	34K	12K	63K
# Cells to annotate	985K	283K	768K
# Unique cells to annotate	264K	138K	379K
Average cell length	20	21	20

In the context of the SemTab2020 challenge, the table corpora are significantly large with thousands of tables and cells to annotate (Table 1). The public Wikidata API is not an ideal lookup service due to restrictions on the number of concurrent connections, result set sizes, and query time. To avoid these limitations, we built our own lookup service using the Wikidata Toolkit[4] and a Spark-based big data platform. Specifically, from the initial raw Wikidata dump, we use the Wikidata Toolkit to filter out the unnecessary documents such as Form, Lexeme, MediaInfo, Sense and Statement documents, so that we only retain entities that are item documents identified by QID and property documents identified by PID, associated with their labels and aliases in all languages. We

[3] In the context of the SemTab2020 challenge, the target cells are provided. In general, DAGOBAH has a module that enables to infer a primitive type for any arbitrary cell so that lookups are only triggered on cells that have entities as values.

[4] https://github.com/Wikidata/Wikidata-Toolkit

then store the filtered dump in a Hadoop Distributed File System (HDFS) and perform the entity lookup via the Spark framework. Table 2 presents the Wikidata KG used in the challenge. It should be noted that during the Round 1, we built the lookup service based on an old version of Wikidata (from 2017) that contains fewer entities (50 millions vs. 86 millions in version 2020), which has resulted in incomplete lookups, and thereby hindered the annotation results as discussed in Section 4.

Table 2. Overview of the Wikidata KG used in the challenge. A 2017 version was used in Round 1. The July 2020 version was used in Round 2, 3 and 4.

Wikidata version	# triples	# entities	# predicates
July 2020	490M	86M	7800
Circa 2017	100M	50M	8200

We provide in Table 3 the complete lookup time for the Round 1, 2, and 3 (line "Current Time"). To construct our lookup service, we considered a pure Python Levenshtein module which has very poor performance (generally, 50 to 100 times slower than the more efficient Cython Levenshtein library[5]). As a result, our current lookup service is not optimized yet, taking about 268h ($\approx$ 11 days) for the lookups of Round 3. In a future system, we will install the Cython Levenshtein library to significantly reduce the execution time. We roughly estimate the potential lookup time of such a future system (line "Ideal Time") by inducing the time of Cython Levenshtein usage from the pure Python Levenshtein's time.

Table 3. Spark Lookup Time (in hours) for Round 1, 2, and 3 using 150 machines. "Current time" is measured using the pure Python Levenshtein module. "Ideal time" is the expected lookup time in a future system using the Cython Levenshtein library.

	Round 1	Round 2	Round 3
Current time	44	64	268
Ideal time	44	36	96

3.2 Candidate Pre-Scoring

The pre-scoring step aims to assign a preliminary confidence score to each candidate $e_c \in \mathcal{E}_c$ generated after the lookup step. The confidence score function (Equation (1)) leverages the semantic contextual relationships and literal similarity between the candidate e_c and the cell e_m.

[5] https://pypi.org/project/python-Levenshtein/

$$Sc(e_c, e_m) = Sc_{context}(\mathcal{N}_{table}(e_m), \mathcal{N}_{graph}(e_c)) * Sc_{sim}(e_m, e_c)^x \tag{1}$$

$Sc(e_m, e_c)$ is the product of a context factor $Sc_{context}(\mathcal{N}_{table}(e_m), \mathcal{N}_{graph}(e_c))$ and a literal factor $Sc_{sim}(e_m, e_c)$. $Sc_{sim}(e_m, e_c)$ gives the highest Levenshtein ratio between the cell and the label set of the candidate. This label set is composed of the labels and aliases of the entity (*e.g.* "France", "La France" for Q142). $x \in \mathbb{N}^+$ allows to define the importance of the textual similarity acting as an amplification factor. We empirically observed that 5 was an appropriate amplification factor for x for this challenge.

The context score is calculated by Equation (2):

$$Sc_{context}(\mathcal{N}_{table}(e_m), \mathcal{N}_{graph}(e_c)) = \begin{cases} \overline{\mathcal{N}_s} \text{ if } \overline{\mathcal{N}_s} \geq 0.1 \\ 0.0001 \text{ otherwise} \end{cases} \tag{2}$$

$\mathcal{N}_{table}(e_m)$ is the set of neighboring cells in the same row as e_m. $\mathcal{N}_{graph}(e_c)$ is the set of neighboring elements of e_c in the KG[6]. Neighboring literals are directly added to $\mathcal{N}_{graph}(e_c)$ whereas, for neighboring entities, their labels and aliases are added. $\mathcal{N}_s$ is a set which contains the best neighborhood matching score ns_i for each neighboring cell $n_i \in \mathcal{N}_{table}(e_m)$ and all neighboring literals or nodes in $\mathcal{N}_{graph}(e_c)$. For each neighboring cell n_i, the neighborhood matching score $ns_i \in \mathcal{N}_s$ is generated as follow:

- If n_i is a string, then ns_i is the highest Levenshtein similarity value.
- If n_i is a number, then ns_i is given by Equation (3):

$$ns_i = \max_{n_g \in \mathcal{N}_{graph}(e_c)} 1 - \frac{|n_i - n_g|}{|n_i| + |n_g|} \tag{3}$$

- If n_i is a date-time value, and if a matching exists between the cell and a neighboring element, then we set ns_i to 1, otherwise $ns_i = 0$.
- If the previous steps gives a ns_i value lower than 0.85, then the system resets ns_i to 0.0001.

For example, the confidence score $Sc(e_m, e_c)$ of the candidate e_c "Q1574185" with the cell e_m "University College Cork" is equal to 1 since "Q1574185" has this exact label and all neighboring cells share information with neighboring elements of "Q1574185".

3.3 Columns-Property Annotation (CPA)

The CPA task involves finding a semantic relation between a pair of ordered columns {head, tail}. In other words, we try to figure out the most suitable relation among the ones connecting a candidate entity in $\mathcal{E}_{head}$ for the head column to a candidate entity in $\mathcal{E}_{tail}$ for the tail column. We employ a simple majority voting strategy which relies on the occurrence and accumulated confidence score of the relation r that we define as:

[6] Neighboring elements are object (resp. subject) of triples whose subject (resp. object) is e_c.

- Occurrence$(r) = \#(e_{head}, e_{tail})$,
- AccumulatedConfidenceScore$(r) = \sum Sc(e_{head}) * Sc(e_{tail})$

such that $e_{head} \in \mathcal{E}_{head}$, $e_{tail} \in \mathcal{E}_{tail}$, and $\langle e_{head}, r, e_{tail} \rangle \in KG$.

In contrast to the relation's head which is always an entity, its tail can be an entity, a textual value, a numerical value, or a date-time value. Literal values may be noisy (*e.g.* "370.1069999997" may represent the integer value "369", "1845-01-01" may correspond to "1845/01/01" or "1845" in the KG). Due to the different natures of each tail type, we consider different matching metrics to verify whether a triple $\langle e_{head}, r, e_{tail} \rangle$ exists in the KG:

- For entity IDs, score$(id_1, id_2) = 1$ if the two ids are exactly the same, otherwise 0.
- For numerical values, score$(num_1, num_2) = 1 - \dfrac{|num_1 - num_2|}{|num_1| + |num_2|}$
- For string values, score$(text_1, text_2) =$ Levenshtein$(text_1, text_2)$
- For date-time values, we compare many variants of date-time values (*e.g.* from the initial value "2020", we could have "2020-01-01" or "2020/01/01"). Two date-time values are matched, *i.e.*, score$(date_1, date_2) = 1$ if one is a variant of the other.

From the criteria above, the relation with the highest occurrence is considered the target relation. If there are several relations having this highest occurrence, we select the one with the highest score. For example, consider the table provided in Fig. 1. The two relations P131 ("*located in the administrative territorial entity*") and P159 ("*headquarters location*") can relate the first (Col 0) and the third (Col 2) columns. Specifically, Q1574185-"University College Cork" (resp. Q245247-"King's College London") is located (P131) in Q36647-"Cork" (resp. Q202059-"London Borough of Lambert"). Cork is also the headquarter (P159) of University College Cork. Therefore, the occurrence count of P131 is 2, and 1 for P159. We conclude that P131 is the CPA for the column pair {Col0, Col2}.

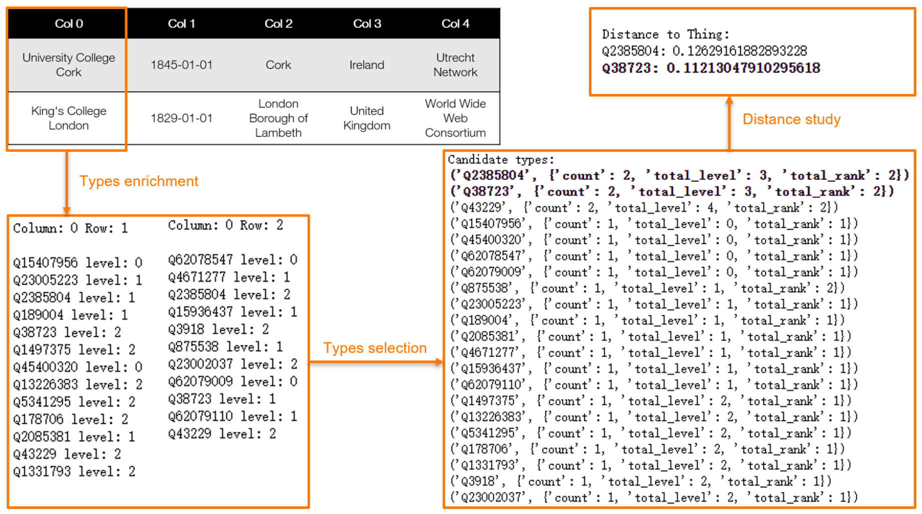
3.4 Cell-Entity Annotation (CEA)

The CEA task aims to annotate the table cells with entities from the KG. Taking the CPA into account, for each candidate entity in a given row, we update its score computed in the pre-scoring step (Section 3.2) by adding a constant score 1 (resp. 0) if the CPA relation can (resp. cannot) connect this candidate to the counterpart in the remaining column. The output entity is the one with the highest score.

For example, for the cell "Cork" in Fig. 1, two of the candidate entities are Q36647 ("Cork city in Munster, Ireland") and Q162475 ("Cork County, Ireland"). They both have the same score output from the pre-scoring step (Section 3.2). Given we have determined the relation between "Col 0" and "Col 2" to be P131 and that the candidate Q36647 is linked to the cell "University College Cork" via this relation, we increase the score of Q36647 by 1. Q36647 then becomes the candidate with the highest score, and is chosen as the CEA output for the "Cork" cell.

3.5 Column-Type Annotation (CTA)

After getting the output of CEA, a majority voting strategy is adapted to identify the most precise type for target columns. This process is illustrated in Fig. 2 for the first column ("Col 0") of the example table depicted in Fig. 1.



Col 0	Col 1	Col 2	Col 3	Col 4
University College Cork	1845-01-01	Cork	Ireland	Utrecht Network
King's College London	1829-01-01	London Borough of Lambeth	United Kingdom	World Wide Web Consortium

```
Column: 0 Row: 1          Column: 0 Row: 2

Q15407956 level: 0        Q62078547 level: 0
Q23005223 level: 1        Q4671277 level: 1
Q2385804 level: 1         Q2385804 level: 2
Q189004 level: 1          Q15936437 level: 1
Q38723 level: 2           Q3918 level: 2
Q1497375 level: 2         Q875538 level: 1
Q45400320 level: 0        Q23002037 level: 2
Q13226383 level: 2        Q62079009 level: 0
Q5341295 level: 2         Q38723 level: 1
Q178706 level: 2          Q62079110 level: 1
Q2085381 level: 1         Q43229 level: 2
Q43229 level: 2
Q1331793 level: 2
```

```
Candidate types:
('Q2385804', {'count': 2, 'total_level': 3, 'total_rank': 2})
('Q38723', {'count': 2, 'total_level': 3, 'total_rank': 2})
('Q43229', {'count': 2, 'total_level': 4, 'total_rank': 2})
('Q15407956', {'count': 1, 'total_level': 0, 'total_rank': 1})
('Q45400320', {'count': 1, 'total_level': 0, 'total_rank': 1})
('Q62078547', {'count': 1, 'total_level': 0, 'total_rank': 1})
('Q62079009', {'count': 1, 'total_level': 0, 'total_rank': 1})
('Q875538', {'count': 1, 'total_level': 1, 'total_rank': 2})
('Q23005223', {'count': 1, 'total_level': 1, 'total_rank': 1})
('Q189004', {'count': 1, 'total_level': 1, 'total_rank': 1})
('Q2085381', {'count': 1, 'total_level': 1, 'total_rank': 1})
('Q4671277', {'count': 1, 'total_level': 1, 'total_rank': 1})
('Q15936437', {'count': 1, 'total_level': 1, 'total_rank': 1})
('Q62079110', {'count': 1, 'total_level': 1, 'total_rank': 1})
('Q1497375', {'count': 1, 'total_level': 2, 'total_rank': 1})
('Q13226383', {'count': 1, 'total_level': 2, 'total_rank': 1})
('Q5341295', {'count': 1, 'total_level': 2, 'total_rank': 1})
('Q178706', {'count': 1, 'total_level': 2, 'total_rank': 1})
('Q1331793', {'count': 1, 'total_level': 2, 'total_rank': 1})
('Q3918', {'count': 1, 'total_level': 2, 'total_rank': 1})
('Q23002037', {'count': 1, 'total_level': 2, 'total_rank': 1})
```

```
Distance to Thing:
Q2385804: 0.12629161882893228
Q38723: 0.11213047910295618
```

Fig. 2. CTA annotation structure

The CTA annotation begins with a type enrichment step. Let $\mathcal{T}_j$ be the set of candidate types for the cell j from a target column. $\mathcal{T}_j$ is represented by a hierarchical tree with 3 levels of types. We consider the types related by the P31 predicate ("*instance of*") to the predicted CEA entity as level 0. Their parent types linked through the P279 predicate ("*subclass of*") are assigned to level 1 and their ancestors to level 2. The system gives priority to the direct types. The rank of a candidate type is another useful factor for the CTA decision. According to Wikidata's mechanism for annotating multiple values for a statement, a type for an entity may have three ranks[7]. We represent those ranks as priority numbers: PREFERRED-2, NORMAL-1 and DEPRECATED-0.

The second step consists in a preliminary selection of types. We compute the frequency, accumulated level, and accumulated rank for all candidate types of a target column, *i.e.* all types appearing in at least one $\mathcal{T}_j$ of the column. We then select the types with the highest occurrence, the lowest accumulated level, and the highest accumulated rank at the same time. In case of ties, we give priority to the occurrence, then to the accumulated level, and finally to the accumulated rank. In the example of Fig. 2, Q2385804 and Q38723 are chosen at this step.

[7] https://www.wikidata.org/wiki/Help:Ranking

The final step consists in computing the distance between the chosen candidate types and the entity Thing (Q35120) in order to select the most specific type. We first consider the mutual inheritance relationship between the remaining candidate types. When such a relationship exists, we select the most specific type. If no inheritance relationship can be found, we compute the shortest distance to Thing. If there is still a tie among some candidate types, the system randomly selects one of them as CTA output.

4 Results and Discussion

Table 4 provides the annotation scores (F1-score, precision, and coverage) of our system for the three tasks in Rounds 1, 2, 3, and 4 of the challenge. Note that the datasets used in Rounds 1, 2, and 3 are automatically generated in a synthetic way by adding some artificial noise (Synthetic Tables). Interestingly, in Round 4, beside a synthetic dataset, a novel corpus (Tough Tables [6]) is introduced and consists of a set of high quality manually-curated tables with complex patterns of cells (*e.g.*, ambiguous names, typos). The results clearly show that a simple, yet optimised, model can achieve very good performance on the SemTab2020 synthetic corpus. This is partly explained by the fact that a row in any tables from this corpus is fully represented in the Wikidata Knowledge Graph (KG). As a consequence, a good lookup service with high coverage and a well-tuned matching strategy are enough to produce very competitive results.

As stated in Section 3, during Round 1, we implemented two variants of DAGOBAH: an embedding-based version (DAGOBAH_Embedding) and an optimised scoring-based version (DAGOBAH_SL). We notice that DAGOBAH_SL performs much better than DAGOBAH_Embedding. The drop in the scores is arguably derived from the fact that DAGOBAH_Embedding retains only 3 clusters for the entity disambiguation, which does not always involve all the good candidates given the imperfection of the current clustering algorithm.

We observe that one difficulty concerns the dynamic and evolving nature of KGs such as Wikidata. During Round 1, we used an outdated KG version (circa 2017 vs. target March 2020, see Table 2). Hence, the lookup service was not able to retrieve relevant candidates for $\approx 5\%$ of the target cells, leading to a significant drop in performance, compared to the leader MTab. Challenge organizers should either consider distributing the reference KG to use alongside the tables to annotate, or consider tables with evolving annotations along the time, anticipating that new items or even properties may be added in the KG.

The other main limitation of Rounds 1, 2, and 3 of SemTab2020 resides in the nature of the data to annotate. Indeed, tables are synthetically generated from the Wikidata knowledge graph. Consequently, the proposed tables are relatively clean and lookup operations can easily match cells in the tables to entities in Wikidata even if noising techniques are introduced. In real-world applications, tabular data can contain complex values (*e.g.*, cells containing lists of entities using various separators, alternative entity names), artifacts (*e.g.*, data encoding problems, formatting errors, input errors, missing values), and more complex lay-

Table 4. Results of the DAGOBAH system in Rounds 1, 2, 3, and 4 of the SemTab2020 challenge. "F1" stands for F1-score, "P" stands for Precision, and "C" stands for Coverage. The coverage is the ratio between the number of annotations proposed by the system and the number of targets to annotate. We also report on the score of MTab as this is the challenge winner.

		CTA			CEA			CPA		
		F1	P	C	F1	P	C	F1	P	C
Round 1 (Synthetic Tables)	DAGOBAH_Embedding	0.779	0.803	95.3%	0.776	0.843	91%	0.809	0.958	73%
	DAGOBAH_SL	0.834	0.854	95.3%	0.922	0.944	95.3%	0.914	0.962	90.6%
	MTab	0.926	0.926	-	0.987	0.988	-	0.971	0.971	-
Round 2 (Synthetic Tables)	DAGOBAH_SL	0.983	0.983	99.9%	0.993	0.993	99.9%	0.992	0.994	99.5%
	MTab	0.984	0.984	-	0.995	0.995	-	0.997	0.997	-
Round 3 (Synthetic Tables)	DAGOBAH_SL	0.974	0.974	99.9%	0.985	0.985	99.9%	0.993	0.994	99.9%
	MTab	0.976	0.976	-	0.991	0.992	-	0.995	0.995	-
Round 4 (Synthetic Tables)	DAGOBAH_SL	0.972	0.972	-	0.984	0.985	-	0.995	0.995	-
	MTab	0.981	0.982	-	0.993	0.993	-	0.997	0.997	-
Round 4 (Tough Tables)	DAGOBAH_SL	0.743	0.745	-	0.830	0.819	-	-	-	-
	MTab	0.728	0.73	-	0.907	0.907	-	-	-	-

outs (*e.g.*, merged rows/columns, multi-line headers, horizontal/vertical/matrix tables) making table manipulation and annotation much more complicated. In order to produce more robust, generic, and intelligent annotation systems, it seems important that evaluation corpora take these challenges into account in the future. An example can be found in the Tough Tables corpus from Round 4 which contains tables manually scraped from the Web. We observe a remarkable degradation in performance of CTA and CEA tasks given the complexity and ambiguity of this corpus.

5 Conclusion and Future Work

In this paper, we presented the improvements implemented in DAGOBAH, a generic and scalable system for table annotation. In particular, our approach involves mutual influences between the two tasks of Cell-Entity Annotation and Columns-Property Annotation to resolve ambiguities. Additionally, by leveraging the type hierarchy, we are able to determine the most fine-grained type in the Column-Type Annotation task. We also focused on the scalability of our system since computational time is an important limitation in real-world use cases.

Our promising results pave the way for several perspectives. First, APIs and GUIs will be developed around DAGOBAH to enable its real-world and industrial usage. Second, following some preliminary work [3], we ambition to leverage embeddings to build an optimized hybrid system, with the intuition that their continuous aspect may help to cope with ambiguities. To this end, we believe that advanced approaches could be motivated and benefit from the availability of more complex or real-world corpora. Such corpora could involve, for example, more ambiguities, noise, or complex layouts [3]. In a broader perspective, such real-world corpora could go beyond the issue of matching tabular data with knowledge graphs to the detection and processing of emergent entities and relations.

References

1. Bhagavatula, C.S., Noraset, T., Downey, D.: TabEL: Entity linking in web tables. In: 14^{th} International Semantic Web Conference (ISWC). pp. 425–441 (2015)
2. Chabot, Y., Grohan, P., Le Calvez, G., Tarnec, C.: Dataforum: Data exchange, discovery and valorisation through semantics. In: 19^{th} French Conference on Extraction et Gestion des Connaissances (EGC). Metz, France (2019)
3. Chabot, Y., Labbé, T., Liu, J., Troncy, R.: DAGOBAH: An End-to-End Context-Free Tabular Data Semantic Annotation System. In: International Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR Workshop Proceedings, vol. 2553, pp. 41–48 (2019)
4. Chapman, A., Simperl, E., Koesten, L., Konstantinidis, G., Ibáñez, L.D., Kacprzak, E., Groth, P.: Dataset search: a survey. The VLDB Journal pp. 1–22 (2019)
5. Cremaschi, M., Avogadro, R., Chieregato, D.: MantisTable: an automatic approach for the Semantic Table Interpretation. In: International Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR Workshop Proceedings, vol. 2553 (2019)
6. Cutrona, V., Bianchi, F., Jiménez-Ruiz, E., Palmonari, M.: Tough tables: Carefully evaluating entity linking for tabular data. In: International Semantic Web Conference. pp. 328–343. Springer (2020)
7. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: 20^{th} ACM International Conference on Knowledge Discovery and Data Mining (KDD). pp. 601–610 (2014)
8. 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 (2017)
9. Fernandez, R.C., Mansour, E., Qahtan, A.A., Elmagarmid, A., Ilyas, I., Madden, S., Ouzzani, M., Stonebraker, M., Tang, N.: Seeping semantics: Linking datasets using word embeddings for data discovery. In: 34^{th} International Conference on Data Engineering (ICDE). pp. 989–1000 (2018)
10. Hassanzadeh, O., Efthymiou, V., Chen, J., Jimnez-Ruiz, E., Srinivas, K.: SemTab2019: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching - Data Sets. Zenodo (2019), https://doi.org/10.5281/zenodo.3518539
11. Ibrahim, Y., Riedewald, M., Weikum, G.: Making sense of entities and quantities in Web tables. In: International Conference on Information and Knowledge Management (CIKM). pp. 1703–1712 (2016)
12. Jimnez-Ruiz, E., Hassanzadeh, O., Efthymiou, V., Chen, J., Srinivas, K.: SemTab 2019: Resources to Benchmark Tabular Data to Knowledge Graph Matching Systems. In: 17^{th} European Semantic Web Conference (ESWC) (2020)
13. Kilias, T., Löser, A., Gers, F.A., Koopmanschap, R., Zhang, Y., Kersten, M.: Idel: In-database entity linking with neural embeddings. arXiv:1803.04884 (2018)
14. Lehmberg, O., Ritze, D., Meusel, R., Bizer, C.: A Large Public Corpus of Web Tables containing Time and Context Metadata. In: 25^{th} International Conference Companion on World Wide Web (WWW Companion). pp. 75–76 (2016)

15. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. In: 36th International Conference on Very Large Data Bases (VLDB). pp. 1338–1347 (2010)
16. Morikawa, H.: Semantic Table Interpretation using LOD4ALL. In: International Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR Workshop Proceedings, vol. 2553 (2019)
17. Mulwad, V., Finin, T., Syed, Z., Joshi, A.: Using linked data to interpret tables. In: 1st International Workshop on Consuming Linked Data (COLD) (2010)
18. Nguyen, P., Kertkeidkachorn, N., Ichise, R., Takeda, H.: MTab: Matching Tabular Data to Knowledge Graph using Probability Models. In: International Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR Workshop Proceedings, vol. 2553 (2019)
19. Oliveira, D., D'Aquin, M.: ADOG: Annotating Data with Ontologies and Graphs. In: International Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR Workshop Proceedings, vol. 2553 (2019)
20. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web **8**(3), 489–508 (2017)
21. Ran, C., Shen, W., Wang, J., Zhu, X.: Domain-specific knowledge base enrichment using wikipedia tables. In: IEEE International Conference on Data Mining (ICDM). pp. 349–358 (2016)
22. Ritze, D., Lehmberg, O., Bizer, C.: Matching HTML Tables to DBpedia. In: 5th International Conference on Web Intelligence, Mining and Semantics (WIMS). pp. 1–6 (2015)
23. Steenwinckel, B., Vandewiele, G., De Turck, F., Ongenae, F.: CSV2KG: Transforming Tabular Data into Semantic Knowledge. In: International Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR Workshop Proceedings, vol. 2553 (2019)
24. Thawani, A., Hu, M., Hu, E., Zafar, H., Divvala, N.T., Singh, A., Qasemi, E., Szekely, P., Pujara, J.: Entity Linking to Knowledge Graphs to Infer Column Types and Properties. In: International Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR Workshop Proceedings, vol. 2553 (2019)
25. Zhang, Z.: Towards efficient and effective semantic table interpretation. In: 13th International Semantic Web Conference (ISWC). pp. 487–502 (2014)