DAGOBAH: An End-to-End Context-Free Tabular Data Semantic Annotation System

Yoan Chabot[1][0000−0001−5639−1504], Thomas Labbé[1][0000−0001−9295−7675], Jixiong Liu[1], and Raphaël Troncy[2][0000−0003−0457−1436]

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

Abstract. In this paper, we present the DAGOBAH system which tackles the Tabular Data to Knowledge Graph Matching (TDKGM) challenge[3]. DAGOBAH aims to semantically annotate tables with Wikidata and DBpedia entities, and more precisely performs cell and column annotation and relationship identification, via a pipeline starting from pre-processing to enriching an existing knowledge graph using the table information. This paper presents techniques for typing columns with fine-grained concepts while ensuring good coverage, and for valuing these types when disambiguating the cell content. This system obtains promising results in the CEA and CTA tasks on the challenge test datasets.

Keywords: Tabular Data · Knowledge Graph · Entity Linking · Embedding · DAGOBAH · TDKGM Challenge

1 Introduction

The annotation of tables using knowledge graphs is an important problem as large parts of both companies internal repositories and Web data is represented in tabular formats. This type of data is difficult to interpret by machines because of the limited context available to resolve semantic ambiguities and the layout of tables that can be difficult to handle. The ability to annotate tables automatically using knowledge graphs (encyclopedia graphs such as DBpedia and Wikidata or enterprise-oriented graphs) allows to support new semantic-based services. For example, it opens the way to more efficient solutions to query (e.g. "moving beyond keyword" for dataset search [2]), manipulate and process heterogeneous table corpus [1].

In this paper, we propose a complete system to annotate tabular data, from pre-processing to entities and relations extracted from knowledge bases, without using any tables context. The main contributions of our system are:

– A new pre-processing tool chain improving the results of the DWTC framework[4].

[3] http://www.cs.ox.ac.uk/isg/challenges/sem-tab/
[4] https://github.com/JulianEberius/dwtc-extractor

- A three-step annotation process (cell-entity annotation, column-type annotation and disambiguation) leveraging on Wikidata and DBpedia.
- An alternative approach based on clustering operations using Wikidata embedding.

2 DAGOBAH: an end-to-end system for annotating tables

DAGOBAH is implemented as a set of tools, used in sequence, to provide three main functionalities.

1. The identification of mapping relationships between tabular data and knowledge graphs.
2. The enrichment of knowledge graphs by transforming into triples the knowledge contained in table. DAGOBAH's target knowledge graph is Wikidata for several reasons including its dynamics, coverage and the quality of data [4]. Thus, adaptations had to be made for the challenge to support DBpedia.
3. The production of metadata that can be used for datasets referencing, search and recommendation processes [1].

To provide these features, DAGOBAH is structured in the following way. The pre-processing modules (Section 2.1) perform the tables cleaning and a first high-level characterisation of their shape and content. The entity linking modules accomplish the tasks of the challenge itself, namely the cell-entity annotation (CEA), the column-type annotation (CTA) and the columns-property annotation (CPA). Two methods have been studied to carry out these tasks: a baseline exploiting lookup and voting mechanisms (Section 2.2) and a geometric approach using a clustering approach based on Wikidata embeddings (Section 2.3).

2.1 Tables Pre-processing

In order to correctly process the information contained in the tables, it is first necessary to infer several characteristics on the shape of the table. In a context of real exploitation in which there is sometimes little or no knowledge on the tables, the information produced by this chain is decisive for the quality of the annotations. The pre-processing toolbox of DAGOBAH, partly based on the DWTC-Extractor, generates four different types of information. The precision of the toolbox was evaluated on round 1 dataset (Table 1) and compared to a modified version of the DWTC (which does not use HTML tags and thus supports more formats).

Table Orientation Detection. The initial algorithm proposed in the DWTC-Extractor is based on the length of the strings and the assumption that the cells in the same column have a similar size. However, the robustness of the DWTC algorithm can be improved. Indeed, for example, two strings representing very different elements may have the same length (for example "Paris" and "$10cm^2$").

DAGOBAH introduces a new algorithm based on a primitive cell typing system with eleven types (string, floating numbers, date, etc.). Based on these types t_i, an homogeneity score is computed on each row and each column x (Equation 1). The mean of all rows and all columns is then compared, and depending on the ratio, the table is said "HORIZONTAL" or "VERTICAL".

$$Hom(x) = [\frac{1}{len(x)} \sum_{t_i \in x} (1 - (1 - 2 * \frac{count(t_i)}{len(x)})^2)]^2 \tag{1}$$

Header Extraction. The algorithm used in DAGOBAH is based on the primitive types defined above. The header extractor assumes that the header of a column contain mainly strings and, in most cases, does not share the type of the column cells. The use of these two heuristics allows to identify if a table contains or not a header with a good accuracy (see Table 1). It should be noted that the DWTC framework also offers a header detection tool but it contains several bugs that make the tool impossible to evaluate.

Key Column Detection. A first step aims to identify a first low level type for each column among five given types (Object (mentions that are potentially lookup-able in a knowledge base), Number, Date, Unit (e.g. "12 km") and Unknown (containing all the unclassified columns)). The key column is an Object column containing a large number of unique values and located on the left side of the table.

Table 1. Precision of pre-processing tasks

Task/Tool	DWTC	DAGOBAH
Orientation Detection	0.9	**0.957**
Header Extraction	Not evaluated	**1.0**
Key Column Detection	0.857	**0.986**

This pre-processing toolbox was especially useful during the first round to automatically identify the information contained in the header as well as the column to be annotated in each table. In addition, when the goal is to enrich a knowledge graph with information contained in tables, key column detection is a critical element in determining the subject of the generated RDF triples. The availability of targets during the second round made the use of this pre-processing chain obsolete. However, this does not affect the usefulness of such tools in real world applications.

2.2 Baseline Approach

Entities Lookup. A preliminary cleaning process is first applied in order to optimize the lookups. The intent is not to correct every string issues, but to have a macroscopic transformation process covering the most known artefacts:

encoding homogenization and special characters deletion (parenthesis, square bracket and non alphanumeric characters). Five lookup services are simultaneously queried to retrieve entity candidates from cell mention: Wikidata API, Wikidata Cirrus Search Engine, DBpedia API, Wikipedia API and an internal ElasticSearch index where Wikidata has been ingested, that contains labels, aliases and types associated to Wikidata QIDs. The benefit of having such source is also to manage the indexes, thus the search strategy. An occurrence count is performed at the output of the lookups to select the most popular candidates, and their corresponding types, among the five services. As DBpedia is the target knowledge base for the challenge, QIDs and Wikidata types are translated to equivalent DBpedia entities (using SPARQL query and following *owl:sameAs* and *owl:equivalentClass* predicates). Ancestors are retrieved for each class in the resulting list through SPARQL query to the DBpedia endpoint in order to have an extended list of candidates types.

Column Typing. Before proceeding to counting, it is necessary to remove non-relevant types. A basic type coverage of the cells criteria is not relevant as right types might be more specific but not frequent enough to be consider as the target ones. To solve this issue, a threshold based on relative scores is used. To each type t in $\{t_i\}$ (list of all types in a given column), a score S_t is first associated, from which a relative score R_t is computed (Equation 2).

$$St = \frac{count(t)}{sum(t_i)} \quad \text{and} \quad Rt = \frac{St}{max(St_i)} \tag{2}$$

Only types with $Rt > 0.7$ (configurable threshold) are considered in the next steps. In order to select the target type from the short-listed ones, a new method called TF-ICF (Type Frequency - Inverse Cell Frequency) is used to compute the specificity. In the spirit of TF-IDF, this score reflects the importance of a type t in $\{t_i\}$ (Equation 3).

$$S_{spec}(t) = S_t * log(\frac{N_R}{count(t)}) \tag{3}$$

where N_R is the number of rows and $count(t)$ is the occurrences of type t within the given column. The advantage of the TF-ICF is to be independent from the target knowledge base.

Entities Disambiguation. In order to enhance the CEA results, the previous ordered types are used to disambiguate the candidates entities. If the first candidate of the lookup has the target type, it is selected as the target entity; if not, we select the entity associated to this type in the lookup list (if the list is empty, no annotation is produced).

2.3 Embedding Approach

The intuition behind this approach is that entities in the same column should be closed in the embedding space as they share semantic characteristics, and thus

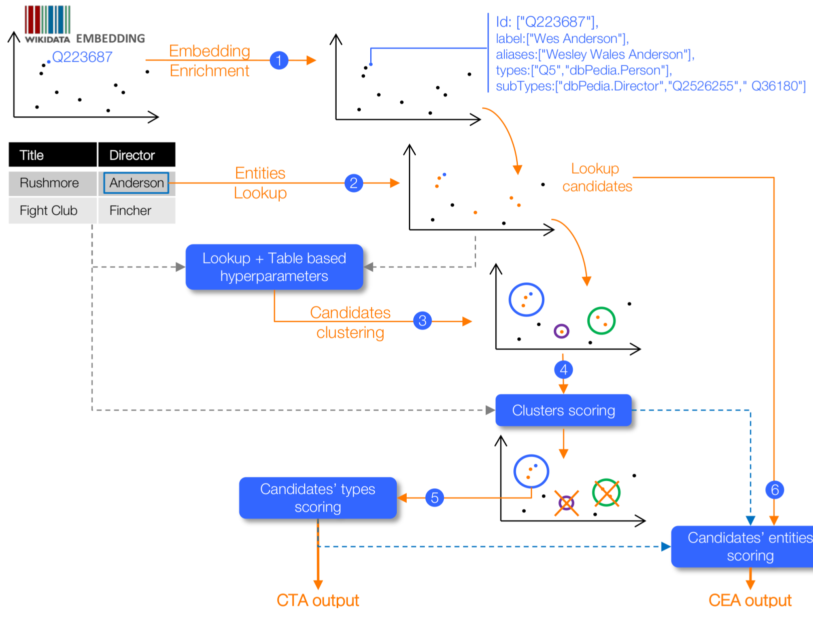

Fig. 1. Workflow of the embedding approach

could form coherent clusters. In order to compare this approach to the baseline, a syntactic lookup to Wikidata embedding is done.

Embedding enrichment and Lookup. The pre-trained Wikidata embedding [5] only contains Wikidata QIDs. Labels, aliases as well as types are added to each entity through an internal Elasticsearch server (step 1 in Figure 1). Lookups are then used to find candidates matching the content of each cell (step 2 in Figure 1). Both regex and Levenshtein distance strategies have been implemented.

1. Entity candidate label or aliases should include all the words of the original mention (with no order); and similarity between the two should be less than a Levenshtein ratio [6] threshold of 0.4 ($item_{sim}(i)$).
2. Levenshtein similarity between an entity string and mention should be larger than 0.9.

If an entity (label or aliases) satisfies one of the previous conditions, it is accepted as candidate.

Candidates Clustering and Scoring. The challenge is to have most of the expected candidates for a given column in one of the clusters. A grid search strategy measuring precision of correct candidates clustering with different algorithms and K values was implemented to determine the best algorithm (K-means with hyperparameter K equals to number of lookup candidates divided by number of rows)(step 3 in Figure 1). In order to select the relevant cluster, a scoring algorithm has been defined (step 4 in Figure 1). The three clusters with the highest rows coverage (i.e. number of rows having at least one candidate in the cluster) are selected. Then, a confidence score is associated to each candidate within these clusters (Equation 4).

$$S_c(i) = item_{conf}(i) * item_{sim}(i) \tag{4}$$

where $item_{conf}(i)$ is the co-occurrence score given by Equation 5.

$$item_{conf} = FE * FH + \sqrt{FT} + T_h \tag{5}$$

$item_{conf}(i)$ is computed from FE, FH and FT defined in Equation 6.

$$FE = \frac{e+1}{N_C} \quad , \quad FH = \frac{h+1}{N_C} \quad , \quad FT = \frac{t}{N_C - 1} \tag{6}$$

where e = number of entities in other columns matching with Wikidata properties values of the candidate; h = number of headers in other columns matching with Wikidata properties labels of the candidate; t = number of exact Wikidata triples of the given candidate row; N_C = number of table columns; T_h = 1 if candidate type matches with the column header, 0 if not. The normalized confidence score for a given cluster n is then computed using:

$$S_k(n) = \frac{\sum_{i \in n} S_c(i)}{len(n)} \tag{7}$$

where $len(n)$ is the number of elements within cluster n.

From all candidates in the 3 clusters selected in step 4, a counting for every existing type is computed, each type inheriting the confidence score of its corresponding candidate (step 5 in Figure 1). All types with a score higher than a threshold ($Max(score) * 0.6$) is selected. Thus, the output type is the one having the highest specificity within DBpedia hierarchy (using subclasses count and distance to $owl : Thing$). Finally, the candidates of each cell resulting from the lookup operations are examined according to the selected type (step 6 in Figure 1). The following score is computed for each lookup entity i belonging to cluster n:

$$S_e(i) = 0.25 * S_k(n) + 0.5 * R_t + 0.2 * S_c(i) \tag{8}$$

where $R_t = 1$ if the entity belongs to the type produced in CTA, 0 if not. From a given row candidates, the output entity is the one with the highest score.

3 Results

The baseline was used during the three rounds of the challenge (Table 2). The CEA's results are satisfactory, but the baseline has difficulty in producing the CTA results expected by the evaluator. In round 1, the predicted type was often too generic or too specific. In addition, the baseline showed two important limitations: a high dependency on lookup services (over which DAGOBAH has

Table 2. Results of the baseline and embedding approaches for the three rounds of the challenge

	Task	CEA		CTA		CPA	
	Criteria	F1 Score	Precision	Prim. Score	Sec. Score	F1 Score	Precision
Round 1	Baseline	0.897	0.941	0.644	0.580	0.415	0.347
	Embedding	TBD	TBD	0.903	0.806	TBD	TBD
Round 2	Baseline	0.713	0.816	0.641	0.247	0.533	0.919
Round 3	Baseline	TBD	TBD	TBD	TBD	TBD	TBD
	Embedding	TBD	TBD	TBD	TBD	TBD	TBD

little control) and difficulties in correctly setting up algorithms (in particular finding the right compromise between specificity and representativeness of types in the case of CTA). Concerning the CPA, a simple lookup technique on the header was used during round 1 explaining the low accuracy.

In round 2, a method of searching for relationships between each pair of instances (output of CEA) of the two candidate columns followed by a majority vote was used. This second technique has significantly improved the accuracy of the CPA.

To show the contribution of the embedding approach, an evaluation was carried out on the corpus of round 1[5]. Tests on the round 1 corpus are currently being finalised with very promising results on CTA. More complete evaluations will be available in the final version of the paper as well as an evaluation of both approaches in round 3. The results presented in the table highlight the embedding approach proficiency to determine the exact type of a column. The embedding approach presents several strengths including the control of lookup strategies (compared to baseline) as well as its ability to give results even on very small tables. In addition, the results are particularly interesting in cases where string mentions in the original table are incomplete. In table 54719588_0_8417197176086756912 for instance, one column references movie directors only by their family names (in that case, our baseline performance was poor). Doing K-means clustering on a subset of this table (four rows) performs pretty well even with very few data, as illustrated in Figure 2 (2 clusters shown among 12). Indeed, the green cluster gives the expected candidates, even if disambiguation still has to be done for some cells.

4 Conclusion and Future Work

In this paper, we have presented a baseline and an embedding approach to carry out the task of generating semantic annotations of tables. The embedding approach shows very encouraging results thanks to its ability to infer candidates from incomplete information. However, optimizing the hyper parameters still remains challenging. The way DAGOBAH computes the number of clusters for

[5] For timing and performance reasons, this approach could not be tested on round 2 in time.

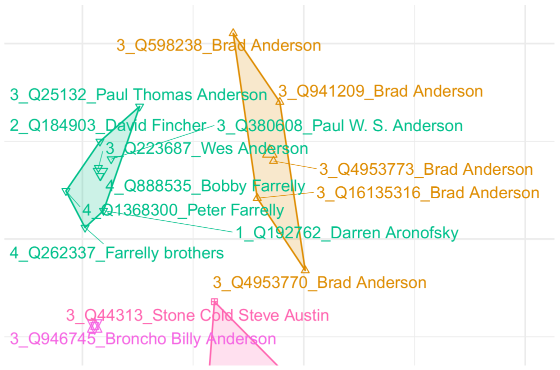

Fig. 2. Result of K-means clustering applied to Wikidata embedding

K-means (based on lookup and table properties) gives good results but might not be robust with all datasets (that is the main reason why we have decided to consider the three high-scored clusters instead of relying entirely on the first one). Other clustering algorithms shall be tested that may be more accurate to find the best compromise between having all candidates in the same cluster and enough clusters for good discrimination between candidates.

We also believe combining Wikipedia and Wikidata in a joint embedding space and use Fasttext-like embeddings, as well as exploiting semantic lookup in addition to syntactic one could significantly enhance the entities mapping. Finally, we are also studying a vectorial approach [3] consisting of learning table rows embedding (possibly Poincaré-like) combined with geometric constraints derived from column mentions, and then find mapping (general or local) transformation(s) with a Wikidata/Wikipedia embedding space.

References

1. Chabot, Y., Grohan, P., Le Calvez, G., Tarnec, C.: Dataforum: Data exchange, discovery and valorisation through semantics. In: Extraction et Gestion des Connaissances (EGC). Metz, France (2019)
2. 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)
3. Chen, J., Jimenez-Ruiz, E., Horrocks, I., Sutton, C.: ColNet: Embedding the Semantics of Web Tables for Column Type Prediction (2018)
4. Färber, M., Ell, B., Menne, C., Rettinger, A.: A Comparative Survey of DBPedia, Freebase, OpenCyc, Wikidata, and YAGO. Semantic Web Journal pp. 1–5 (2015)
5. Han, X., Cao, S., Lv, X., Lin, Y., Liu, Z., Sun, M., Li, J.: Openke: An open toolkit for knowledge embedding. In: International Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 139–144 (2018)
6. Sarkar, S., Pakray, P., Das, D., Gelbukh, A.: JUNITMZ at SemEval-2016 Task 1: Identifying semantic similarity using levenshtein ratio. In: 10^{th} International Workshop on Semantic Evaluation (SemEval). pp. 702–705 (2016)