MTab4Wikidata at SemTab 2020: Tabular Data Annotation with Wikidata

Phuc Nguyen[1], Ikuya Yamada[3], Natthawut Kertkeidkachorn[2], Ryutaro Ichise[1,2], and Hideaki Takeda[1]

[1] National Institute of Informatics, Japan
[2] National Institute of Advanced Industrial Science and Technology, Japan
[3] Studio Ousia, Japan

Abstract. This paper introduces an automatic semantic annotation system, namely MTab4Wikidata, for the three semantic annotation tasks, i.e., Cell-Entity Annotation (CEA), Column-Type Annotation (CTA), Column Relation-Property Annotation (CPA), of Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2020). In particular, we introduce (1) a novel fuzzy entity search to address misspelling table cells, (2) a fuzzy statement search to deal with ambiguous cells, (3) a statement enrichment module to address the Wikidata shifting issue, (4) an efficient and effective post-processing for the matching tasks. Our system achieves impressive empirical performance for the three annotation tasks and wins the first prize at SemTab 2020. MTab4Wikidata is ranked 1st in the two tasks of CEA and CPA, and 2nd rank in the CTA task on the round 1, 2, 3 datasets and 1st rank on the round 4 dataset and the Tough Tables (2T) dataset.

1 Introduction

Thanks to the Open Data movements, many tabular resources have been made available on the Web and data portals. However, it is difficult to use such tabular data because of missing or incomplete metadata, heterogeneous table schema, table cell ambiguity, or misspelling. A promising solution to improve these tabular data's usability is to generate semantic annotations for table elements using knowledge graph concepts. As a result, such annotations could be useful for other downstream tasks such as data management and knowledge discovery.

This paper introduces an automatic semantic annotation system, namely MTab4Wikidata, to match table elements into Wikidata concepts. This system is particular designed for the Semantic Web Challenge's annotation tasks on Tabular Data to Wikidata Matching (SemTab 2020). Figure 1 depicts the three annotation tasks including Cell-Entity Annotation (CEA), Column-Type Annotation (CTA), Column Relation-Property Annotation (CPA). The SemTab 2020 could be formalized as follows.

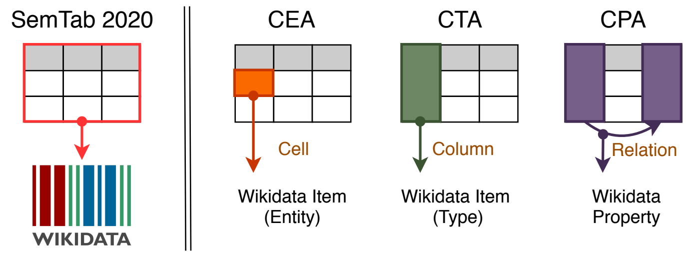


Fig. 1. Semantic Web Challenge on Tabular Data to Wikidata Matching

Wikidata Knowledge Graph: Given Wikidata as a knowledge graph of $G = (Q, P, S)$, where Q and P are the set of Wikidata items, Wikidata properties, respectively. A Wikidata item e ($e \in Q$) is represented with a unique identifier as Q###; for example, Q2 is the "Earth" item. The P### unique identifier represents a Wikidata property p ($p \in P$); for example, P1082 is for the "population" property. S is a set of statements represented as a triple format of Subject - Predicate - Object. These statements describe Wikidata item information such as literal information or connect Wikidata items[1]. For example, the Q5 (Earth) item could have a literal statement as Q5 (Earth) - P1082 (population) - 7,655,957,369; and a item statement as Q5 (Earth) - P276 (location) - Q7879772 (inner Solar System).

Tabular Data: Let a table T be a two-dimensional tabular structure consisting of an ordered set of N rows and M columns. n_i is a row of a table ($i = 1...N$), m_j is a column of the table ($j = 1...M$). The intersection between a row n_i and a column m_j is $c_{i,j}$ is a value of the cell $T_{i,j}$.

Annotation Tasks: We formalize the tabular data to Wikidata annotations as the three matching tasks as follows.

- Cell-Entity Annotation (CEA): Matching a table cell $c_{i,j}$ into a Wikidata item (entity) e.
$$c_{i,j} \xrightarrow{\text{CEA}} Q \tag{1}$$
- Column-Type Annotation (CTA): Matching a column m_j into a Wikidata item (type) e.
$$m_j \xrightarrow{\text{CTA}} Q. \tag{2}$$
- Column Relation-Property Annotation (CPA): Matching the relation between two columns m_{j_1} and m_{j_2} ($j_1, j_2 \in [1, M], j_1 \neq j_2$) to a relation p.
$$r_{m_{j_1}, m_{j_2}} \xrightarrow{\text{CPA}} P \tag{3}$$

[1] Wikidata has about 89.75 million items, 7,972 properties, and 1.14 billion statements in October 2020.

SemTab 2020 Challenges SemTab 2020 tasks are challenging for many reasons as follows.

Tabular data difficulties:

- Tabular data does not have metadata to describe the semantic meaning of table elements.
- Table headers are ambiguous; for example, the most popular table headers are "col0", "col1", and "col2"; therefore, it is hard to understand table schema if we only rely on the headers.
- Table cells are ambiguous, contain many spelling errors and abbreviations. As a result, directly performing entity search with cell values (e.g., "Tokyo") using standard APIs (e.g., Wikidata API search[2] or Wikidata query[3]) could return whereas too many relevant entities or no relevant entities in the responding list. For example, searching with Wikidata API for an ambiguous cell value of "Tokyo" could get 15,780 relevant entities, while there are no relevant entities for a misspelling cell of "6C 124133+40580".

Knowledge graph difficulties:

- Noisy Schema: Wikidata is a data-oriented knowledge graph. It has many entity types annotated by humans; therefore, these types are noisy and not consistent for many Wikidata items. As a result, Wikidata schema standardization is still an open challenge.
- Wikidata Shifting: Wikidata is a fast-evolving knowledge graph, with around 13 million edited Wikidata items per month[4]. According to our analysis on the Round 1 data (Wikidata from March to August 2020), 56.09% of the matching items had changed their information statements (possibly affect the CEA and CPA tasks), and 4.49% changes are related to Wikidata item types (possibly effect on the CTA task).

Related work SemTab 2019 is the last year's challenge on tabular data to DBpedia matching [1]. This year's challenge is tabular data to Wikidata matching; it has a new set of difficulties such as larger-scale knowledge graph setting, knowledge graph data shifting, and noisy schema structure of Wikidata. Additionally, this year's challenge also has a more challenging dataset (the tough tables [2]), which is manually curated, offering realistic issues than the last challenge.

The last winner of SemTab 2019 is MTab system [3] based on an aggregation of multiple cross-lingual lookup services and probabilistic graphical model.

[2] Wikidata search: https://www.wikidata.org/w/api.php
[3] Wikidata query: https://query.wikidata.org/
[4] https://stats.wikimedia.org/#/wikidata.org/content/edited-pages/normal—line—2-year— total—monthly

CSV2KG (IDLab) also uses multiple lookup services to improve matching performance [4]. Tabular ISI implements the lookup part with Wikidata API and Elastic Search on DBpedia labels, and aliases [5]. ADOG [6] system also uses Elastic Search to index knowledge graph. LOD4ALL first checks whereas there is available entity which has a similar label with table cell using ASK SPARQL, else perform DBpedia entity search [7]. DAGOBAH system performs entity linking with a lookup on Wikidata and DBpedia; the authors also used Wikidata entity embedding to estimate the entity type candidates [8]. Mantis Table provides a Web interface and API for tabular data matching [9].

2 Approach

This section describes MTab4Wikidata's assumptions (Section 2.1) and the overall framework in Section 2.2.

2.1 Assumption

Assumption 1 *MTab4Wikidata is built on a closed-world assumption.*

Assumption 2 *Input tables are in the vertical relation type.*

Assumption 3 *The table cells in a column have the same entity types and data types.*

Assumption 4 *The first row of the table (n_1) is the table header. The first cell of column is header of this column, $c_{1,j} \in m_j$.*

Assumption 5 *The first column of the table (m_1) is the core attribute where a cell value in this column could be matched into a Wikidata item, and other remaining cells in the same row could be matched into the item statements.*

2.2 Framework

MTab4Wikidata is an automatic annotation system that could address the three annotation tasks of CEA, CTA, and CPA. Unlike our previous system MTab (DBpedia matching) built on the joint probability distribution of table element signals[3], this work focus on improving the entity search capability as follows:

- Wikidata shifting: Since all history revisions (entity labels, statements) are available at Wikidata item history revision, and we use those revisions to enrich the Wikidata dump; to this end, enhance matching performance.
- Table cell noisiness and misspelling: Due to a high level of spelling mistakes in table cells, we build a novel fuzzy entity lookup using edit distances that could handle fuzzy queries.

– Table cell ambiguity: We also build a novel fuzzy statement search that could find relevant statements from two cells taken from a table. This searcher is built based on the assumption that there is an existing logical relation occurring in the knowledge graph between the table's two cells. The searcher helps reduce the number of relevant candidates for ambiguous queries.

MTab4Wikidata contains three steps pipeline as Figure 2. Step 0 is to prepare Wikidata resources with the entity dump and Wikidata history revisions. We use these resources to create the two indexes of fuzzy entity search and fuzzy statement search. In Step 1, we perform table cells lookup to find relevant entity candidates for each table cell (using the fuzzy entity search) and the two cells in the same row (using the fuzzy statement search). In Step 3, we perform a post processing with a value matching module, then majority voting to select the CEA, CPA, and CTA annotations. The details of each step are described in Section 2.3 (Step 0), Section 2.4 (Step 1), and Section 2.5 (Step 2).

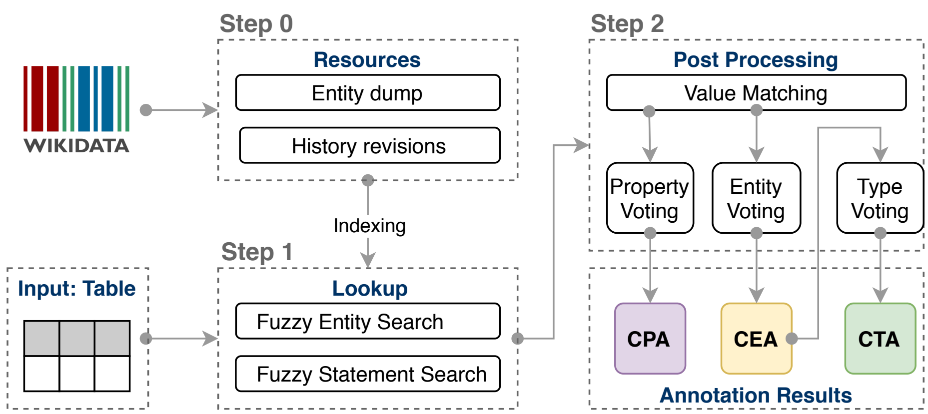


Fig. 2. MTab4Wikidata framework

2.3 Step 0: Wikidata Processing

In this module, we perform the following operations:

– Extracting Wikidata items and these statements: We extract this information from the Wikidata 29 June 2020 entity dump. Note that this entity dump is not available on the Wikidata server now, since they only keep the two months' latest dumps.
– Getting history revision of Wikidata items: We crawl Wikidata item history revisions as a statement form (Subject - Predicate - Object) from 1 March to 29 June 2020[5]. We use those revisions to enrich the Wikidata

[5] Revision history: https://www.mediawiki.org/wiki/API:Revisions.

dump statements. Since we do not know when the challenge's tabular data is generated, we only use the adding operation for all history statements of Wikidata items. In other scenarios, where we know the exact time point of Wikidata, we can reconstruct Wikidata items at this time point by using adding, removing, modifying operations.

– Building an entity index: We build this index using a hash table with the entity dump data using multilingual labels, aliases, and identifiers (the identifiers to other sources). In total, the fuzzy entity search index about 150 million entity names for 87 million items of Wikidata.
– Building a statement index: we use a sparse matrix to build an index for Wikidata statements. To be simple, we only encode the item statements (item - property - item) and ignore the statements' property information. In the general case, we can also index the literal statements and properties into the sparse matrix. Overall, we index around 500 million statements.

2.4 Step 1: Lookup

In this step, we perform entity candidate generation for each target cells using the fuzzy entity search, and two target cells in the same rows using the fuzzy statement search. Figure 3 depicts the semantic annotations of the table of 00KKIZPQ in the round 3 data. We use this table as an example to describe the two following searching module.

col0	col1	col2	col3
V*!AY Psc	-4.593	Pisces	24.280
SDSS J153509.57+360054.5	-17.028	Boötes	232.909

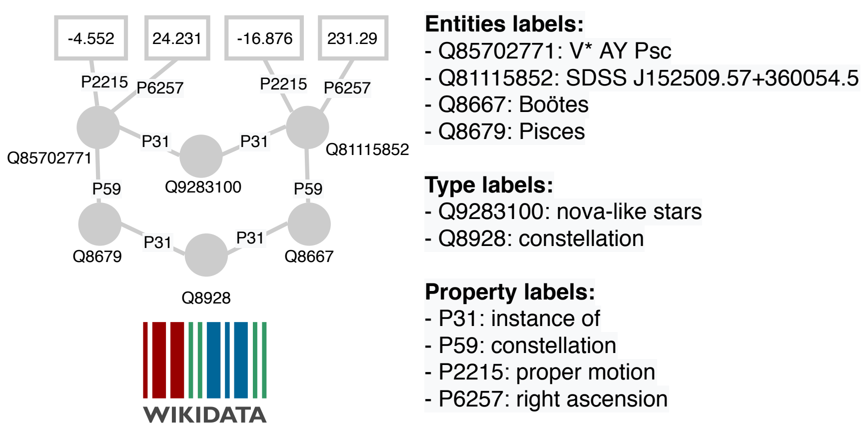


Fig. 3. Semantic Annotations of the table of 00KKIZPQ in the round 3 data

A cell search In this searcher, we perform a fuzzy entity search on the entity index. Given a table, this searcher returns a ranking list of relevant entities ranked with an edit distance, specifically Levenshtein distance. We first start with two edits; if the searcher does not have answers, the number of edits is gradually increased to a maximum of six edits. For example: in Table 3, we have four queries as two misspelling queries: "V***!**AY Psc", and "SDSS J15**3**509.57+360054.5". This fuzzy search returns the correct answers for the two misspelling queries as Q85702771 (V* AY Psc) and Q81115852 (SDSS J152509.57+360054.5). This searcher achieves coverage of 99.89% on average when performing entity lookup for table cells on SemTab 2020 datasets.

Two cells search We design this searcher to handle the ambiguity of table cells. We assume a logical relation equivalent to a statement between two cells of a table row. Therefore, we perform the two cells' search on the target cells in the same rows. The first cell is in the core attribute column, and the second cell is in the remaining columns. We ignore the rows do not have enough two target cells. This idea is inspired by our novel idea of entity-entity column matching in MTab [3]. Unlike our original idea in MTab that performs the two cells matching in the post-processing step, this work adapts this idea into the lookup step and makes the post-processing more efficient. Using statement search in the lookup step is also efficient because of the sparse matrix searching. This idea also works effectively, and robust on DBpedia [10].

For example: in Table 3, we have two queries: the two cells of row 1 "V***!**AY Psc", "Pisces", and row 2 "SDSS J15**3**509.57+360054.5", "Boötes". We first search one cell then check all combinations between the two respond lists, whereas the statements are available. In the end, we only keep entity candidates at two cells that have a statement available. Regarding the cells in the core attribute column, we concatenate all the first cell response candidates.

Finally, for each cell that need to be annotated, we priority select the entity candidates from two cells search. If we do not have candidates in the "two cells" searches, we select the candidates from a cell search.

2.5 Step 2: Post processing

We perform value matching of the cells in the core attribute with the remaining cells in their corresponding rows in the post processing step. This module performs context similarity calculation between candidate statements and table row values then ranks the entity candidates based on these similarities.

Value matching This module applies to each data row of a table. We calculate the similarities between the candidate statements in the core attribute column with other corresponding values in the same row for each table's data row. If many statements have the same similarity score, we select all these statements. We calculate the context similarity of the candidate in the core attribute by taking the average of all statement similarities in the same row.

Figure 4 depicts an example of value matching for a row in the table of 00KKIZPQ Round 3. Recall the cell “V*!AY Psc” has two entity candidates with the ID of Q85702771, and Q86769669. We calculate the statement similarities of each candidate and the remaining cells in this row. For example, we found three similar statements to the cell values in candidate 1, while only found one statement similar in candidate 2. As a result, candidate 1 gets higher context similarity than candidate 2.

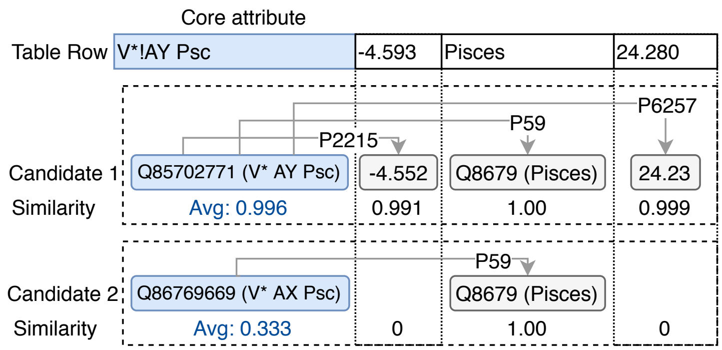


Fig. 4. Value matching for a row in table 00KKIZPQ of Round 3

Annotation

CEA: If the target cells are in the core attribute, we select a candidate with the highest context similarity as a CEA annotation. If the target cells are not in the core attribute, we infer these annotations from CEA results in the core attribute using value matching similarities.

CPA: To get the CPA annotations, we aggregate all properties of statement candidates in the same rows, then using majority voting to select the CPA annotations.

CTA: To get the CTA annotation, we get the direct types from the CEA annotations in a column and vote for the majority types to get CTA annotations.

In the CPA task and CTA tasks, if the system returns many answers for one target annotation, we randomly select one answer as the final annotation.

3 Results

SemTab 2020 has four rounds, where round 1, 2, 3 were implemented as a form of AICrowd leaderboard, and round 4 was implemented as a blind setting. There are five datasets in total, including the four-round datasets and the Tough Tables (2T) dataset [6]. The four rounds datasets were generated by an automatic dataset generator [1], while the 2T dataset is manually curated for table annotations [2].

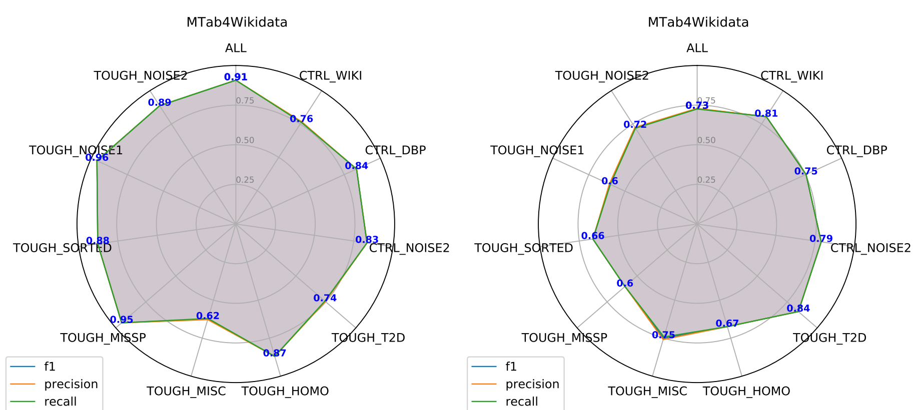
[6] The 2T dataset is in the round 4 dataset

Table 1. Overall result of MTab4Wikidata system on the four round datasets and the tough tables (2T) at SemTab 2020

Dataset	CEA			CTA			CPA		
	F1	Precision	Rank	AF1	Precision	Rank	F1	Precision	Rank
Round 1	0.987	0.988	1	0.885	0.884	2	0.971	0.991	1
Round 2	0.995	0.995	1	0.984	0.984	2	0.997	0.997	1
Round 3	0.991	0.992	1	0.976	0.976	2	0.995	0.995	1
Round 4	0.993	0.993	1	0.981	0.982	1	0.997	0.997	1
2T	0.907	0.907	1	0.728	0.730	1	-	-	-
Average	0.974	0.975	1	0.911	0.911	1.6	0.990	0.995	1

Table 1 reports the overall results of MTab4Wikidata for three matching tasks (CEA, CTA, and CPA) in SemTab 2020 datasets (the four datasets of four rounds and the 2T dataset). Overall, these results show that MTab4Wikidata achieves impressive performances for the three matching tasks: the 1st rank in the two tasks of CEA and CPA, and the 2nd rank in the CTA task in round 1, 2, 3 and the 1st rank in round 4 and the Tough Tables (2T) dataset.

The details of MTab4Wikidata's performance on the 2T dataset is shown in Figure 5. The overall performance decrease in both tasks of CEA and CTA compared to the four-round datasets. Although the tough tables dataset is more difficult with a high level of table noisiness, our system still performs effectively. The F1, Precision, and Recall are quite similar in our results because our system tries to make as many annotations as possible. Even though we do not have a corresponding entity in the knowledge graph, we also return the most relevant entities.



(a) **CEA** (b) **CTA**

Fig. 5. MTab4Wikidata results on the Tough Tables dataset

4 Conclusion

This paper presents an automatic annotation system called MTab4Wikidata to match tabular elements: i.e., cells, columns, relations between columns to Wikidata concepts. In this work, we focus on improving the standard entity search with the fuzzy search and fuzzy statement search to deal with misspelling and ambiguity of table content. The fuzzy search works effectively and achieves an average of 99.89% recall on the three rounds of SemTab 2020. The statement search also gives a tremendous efficient improvement where it could eliminate non-statements candidates. To this end, MTab4Wikidata wins the first prize at SemTab 2020.

Acknowledgements This work was supported by the Cross-ministerial Strategic Innovation Promotion Program (SIP) Second Phase, "Big-data and AI- enabled Cyberspace Technologies" by the New Energy and Industrial Technology Development Organization (NEDO).

References

1. Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, and Kavitha Srinivas. Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems. In *ESWC*, pages 514–530. Springer, 2020.
2. Vincenzo Cutrona, Federico Bianchi, Ernesto Jiménez-Ruiz, and Matteo Palmonari. Tough tables: Carefully evaluating entity linking for tabular data. In *ISWC*, pages 328–343. Springer, 2020.
3. Phuc Nguyen, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. Mtab: Matching tabular data to knowledge graph using probability models. In *SemTab@ ISWC*, pages 7–14, 2019.
4. Gilles Vandewiele, Bram Steenwinckel, Filip De Turck, and Femke Ongenae. Cvs2kg: Transforming tabular data into semantic knowledge. In *SemTab@ ISWC*, pages 33–40, 2019.
5. Avijit Thawani, Minda Hu, Erdong Hu, Husain Zafar, Naren Teja Divvala, Amandeep Singh, Ehsan Qasemi, Pedro A Szekely, and Jay Pujara. Entity linking to knowledge graphs to infer column types and properties. In *SemTab@ ISWC*, pages 25–32, 2019.
6. Daniela Oliveira and Mathieu d'Aquin. Adog-annotating data with ontologies and graphs. In *SemTab@ ISWC*, pages 1–6, 2019.
7. Hiroaki Morikawa. Semantic table interpretation using lod4all. In *SemTab@ ISWC*, pages 49–56, 2019.
8. Yoan Chabot, Thomas Labbe, Jixiong Liu, and Raphaël Troncy. Dagobah: an end-to-end context-free tabular data semantic annotation system. In *SemTab@ ISWC*, pages 41–48, 2019.
9. Marco Cremaschi, Roberto Avogadro, and David Chieregato. Mantistable: an automatic approach for the semantic table interpretation. In *SemTab@ ISWC*, pages 15–24, 2019.
10. Phuc Nguyen, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. Tabeano: Table to knowledge graph entity annotation. *arXiv preprint arXiv:2010.01829*, 2020.