Semantic Table Interpretation using LOD4ALL

Hiroaki Morikawa

Fujitsu Laboratories Limited, 4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki, Kanagawa, Japan

Abstract. In this paper, we describe Semantic Table Interpretation using LOD4ALL. LOD4ALL is an LOD search engine developed by Fujitsu laboratories. This engine crawls Linked Open Data from the Web and provides a high-speed search service. There are many tabular data on the Web, and these data are important sources of Knowledge Graphs. Therefore, we have enhanced a crawler that is a component of LOD4ALL for taking in these tabular data. This crawler is able to construct Knowledge Graphs to a tabular data. To evaluate of the function of this crawler, we have participated in the challenge "Semantic Web Challenge on Tabular Data to Knowledge Graph Matching".

1 Presentation of the system

1.1 General statement

There are a great number of tabular data on the Web[1]. It is useful to build Knowledge Graphs from these tabular data, and we can maintain the latest Knowledge Graph by constructing it to tabular data and importing it[2]. Fujitsu laboratories developed LOD4ALL[3] in 2014, is the world's first repository enabling unified access to Linked Open Data (LOD) through a single query to the entire LOD datasets. Therefore, we have enhanced a crawler that is a component of LOD4ALL for taking in these tabular data. This crawler has the function of constructing Knowledge Graphs to tabular data. To evaluate the function of this crawler, we have participated in the challenge "Semantic Web Challenge on Tabular Data to Knowledge Graph Matching"[4]. In this paper, we describe our proposed approach to realizing this function of our crawler.

1.2 Specific techniques used

In this section, we describe the overview of our proposed approach in the following five steps:

Step 1 Extract candidate entities

Step 2 Resolve an entity type
Step 3 Determine a column type
Step 4 Determine an entity for each cell
Step 5 Extract a relationship of entities for each row

Our proposed approach is similar to that of Efthymiou et al.[5], and, we have imitated the approach of Zwicklbauer et al.[6] from Step 1 to Step 3, and we have enhanced Step 3 in our approach. Step 4 and Step 5 are original processes. Step 0 builds some databases as a preparatory step. Fig. 1 is an overview of our proposed system.

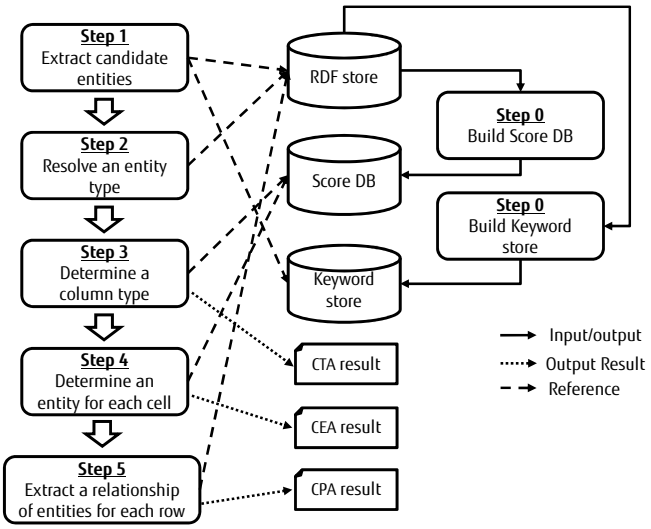

Fig. 1. Overview of our proposed system

Step 0: Build Databases Step 0 builds some databases for a predict on. Our proposed approach can only build these from a target Knowledge Graph. Therefore, our approach does not build predicted databases using annotated tables like T2Dv2[1]. Our approach has two databases. One is the Score DB to resolve CTA task in the challenge. Another is the Keyword Store for Step 1.

In DBpedia, an entity has some classes. For example, `dbr:Barack_Obama` has `dbo:Person`, `dbo:Politician`, and `dbo:President`. For recognizing a more detailed class, we have adopted Okapi BM25[7], which is a traditional retrieval score as the Score DB. To calculate this score, we have considered entities as terms and classes as documents. Adopting this score to DBpedia, the Okapi BM25 score of `dbr:Barack_Obama` is Table 1. By using the Okapi BM25, we could be assigned a higher score to a more detailed class.

[1] http://webdatacommons.org/webtables/goldstandardV2.html

Table 1. The Okapi BM25 score of dbr:Barack_Obama

Class	Score
dboPerson	0.2258
dboPolitician	1.2852
dboPresident	5.1031

Step 1: Extract candidate entities This step obtains candidate entities for each cells in a target tabular data. We have used the literal search function of LOD4ALL. The original literal search function of LOD4ALL uses Elasticsearch that returns a subject (a candidate entity) corresponding to an object that matches a cell value. We have enhanced this literal search function for this challenge. This function can obtain candidate entities in the following steps.

Step 1-1 Direct search: First, by combining `http://dbpedia.org/resource/` with the cell value, we create a candidate entity. Next, we execute ASK query to RDF store. For example, if the cell value is "Japan", executing SPARQL like ASK { `<http://dbpedia.org/resource/Japan>` ?p ?o . }.

Step 1-2 Keyword search: We have enhanced the literal search function of LOD4ALL that can retrieve subjects that have `rdfs:label`, `foaf:name`, `foaf:surname` and `foaf:givenName` in the triple. Further more, we have added keywords for a person that combines a first character, the object value corresponding to `foaf:givenName`, "." and the object value corresponding to `foaf:surname`. Through this enhancement, we can retrieve `dbr:Barack_Obama` from the cell value of "B. Obama". In addition, a similar string search step has been added. We have adopted SimString[8] for the similar string search.

Step 1-3 Output candidate entities: To Output in order of highest TOP-K score. We have set 100.0 for the score of an entity found by Direct search, and set a value multiplying the Elasticsearch's score and the SimString's score for the score of an entity that have found by Keyword search.

The flow of this step is Fig. 2

Step 2: Resolve an entity type In this step, classes for each entity are obtained from the Score DB. This process is similar to those in to previous studies.

Step 3: Determine a column type In this step, estimating a column type (CTA result) using candidate entities extracted in Step 2 and the Score DB. This step consists of the three following steps:

Step 3-1 Calculate a cover ratio

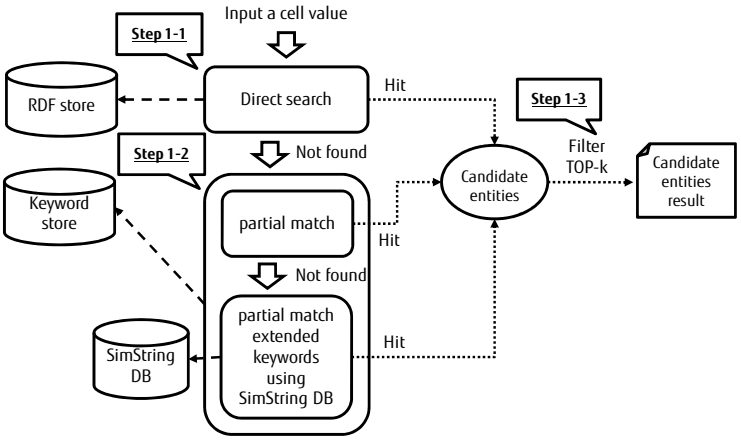

Fig. 2. The flow of extraction of candidate entities

Step 3-2 Search a class score from the score DB
Step 3-3 Calculate a predict score using both a score ratio and a class score

We illustrate the running example of Step 2 and Step 3 using a part of the "List_of_residences_of_Presidents_of_the_United_States#Summer_White_House#0.csv" in Round2 in Fig. 3. We have set Step 1's K as 1 in this figure to explain easily. In this figure, our approach outputs `dbo:President` as the result of this step.

Step 4: Determine an entity for each cell In this step, we determine an entity for each cell that has the class (the column type) determined in Step 3 from candidate entities that extract by Step 1. The output of this step is the result of the CEA task.

Step 5: Extract a relationship of entities for each row In this step, we first collect candidate predicates using entities determined in Step 4 by executing SPARQL (Listing 1.1) to RDF store. %URI1% and %URI2% in SPARQL are replaced entities determined in Step 4. Next, by calculating a frequency, we obtain the predicate of the greatest frequency. The output of this step is the result of the CPA task.

We illustrate the running example of the Step 5 in Fig. 4.

Listing 1.1. SPARQL query used the Step 5

```
PREFIX dbr:   <http://dbpedia.org/resource/>

select distinct ?predicate where{
    %URI1% ?predicate %URI2%
}
```

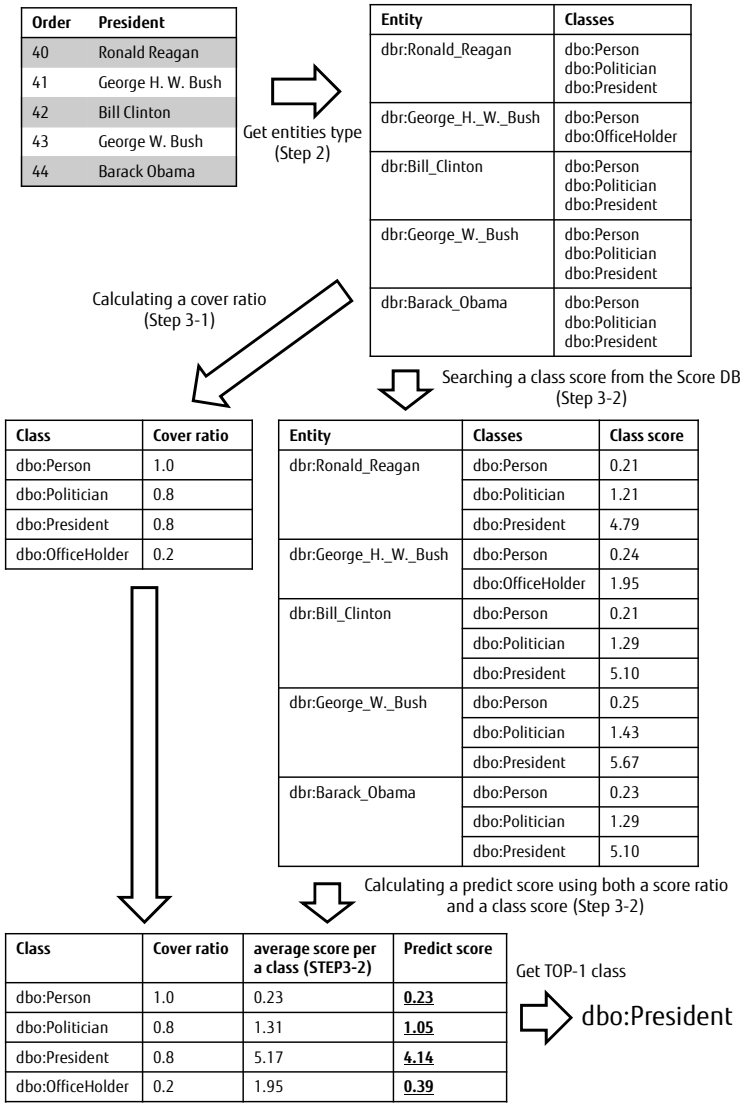

Fig. 3. The running example of Step 2 and Step 3

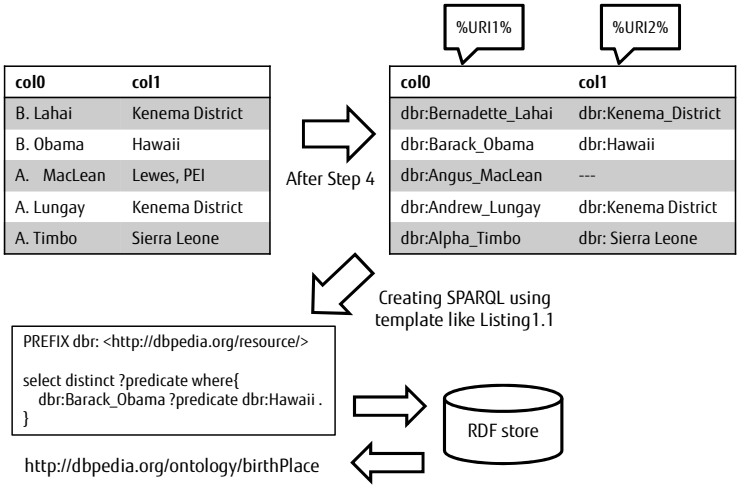

Fig. 4. The running example of Step 5. %URI1% and %URI2% in SPARQL are replaced entities, like this figure.

1.3 Adaptations made for evaluation

We designed predict score functions for Step 3 through trial and error. The best result in our trial is Equation 1. Here, the $normalizedClassScore$ is the class score that has scaled between 0 and 1. α and β are hyperparameters.

$$PredictScore = \alpha * normalizedClassScore + \beta * ratioScore \quad (1)$$

In our experiment, by setting α=1.2 and β=2.5, the system has output the best of results. In Step 1, the top k number is K=10, and the similarity parameter for SimString is 0.7.

1.4 Link to the system and parameters file

We plan to publish our modules in the GitHub[2] .

2 Results

Table 2 is the results of the challenge.

There were many tables with column data for person name labels in the Round4 dataset. Therefore we enhanced the entities lookup for person name in Step 1 after closing Round4. As the results, we were able to improve these scores. We found that their accuracy depended heavily on the accuracy of the entity lookup function in Step 1.

[2] https://github.com/lod4all/semanticTableInterpretation

Table 2. Results of the challenge

		AH-Score	AP-Score	F1-Score	Precision	Leaderboard's ranking
Round1	**CTA**	-	-	0.850	0.850	4/13
	CEA	-	-	0.852	0.874	6/11
	CPA	-	-	-	-	-
Round2	**CTA**	0.893	0.234	-	-	5/9
	CEA	-	-	0.757	0.767	6/10
	CPA	-	-	0.555	0.941	4/7
Round3	**CTA**	1.442	0.260	-	-	5/11
	CEA	-	-	0.828	0.833	6/9
	CPA	-	-	0.545	0.853	5/9
Round4	**CTA**	1.071	0.386	-	-	6/8
	CEA	-	-	0.648	0.654	7/9
	CPA	-	-	0.439	0.904	6/9
Round4 (After closing)	**CTA**	1.41	0.369	-	-	-
	CEA	-	-	0.815	0.818	-
	CPA	-	-	0.459	0.918	-

3 General comments

In these challenge datasets, there were several cases where it was difficult to determine the column type in the CTA task. One of them is shown in Fig. 5. This table is the ranking of Forbes Korea Power Celebrity in 2012[3]. We found this in the Round2 dataset. The Name column in this table may be assigned `dbo:Person` and `dbo:Group`. It is difficult to determine a unique column type for CTA task. To deal with this case, we believe it is necessary to ues an evaluation method that allows multiple perfect annotations for a column.

4 Conclusions

In this paper, we described Semantic Table Interpretation using LOD4ALL. Our proposed approach is still in the early stages, so there are a lot of problems to be resolved. In the future, we will improve the accuracy of collecting candidate entities in Step 1 and the prediction process in Step 3 in these problems in particular. Further more, we will improve our system by developing an ensemble approach between our approach and others. For CTA task, we believe it is necessary to improve DBpedia. For example, we will assign a class to an entity that has not been assigned a class by adopting Fang's method[9].

[3] https://en.wikipedia.org/wiki/Forbes_Korea_Power_Celebrity

Rank	Name	Profession
1	Girls' Generation	Girl group
2	Big Bang	Boy band
3	IU	Singer
4	Kara	Girl group
5	Kim Yuna	Figure skater
6	Lee Seung-gi	Singer, actor, TV show host
7	Park Ji-sung	Footballer
8	Kim Tae-hee	Actress
9	Beast*	Boy band
10	Park Tae-Hwan	Swimmer

Fig. 5. An difficult example of determining a unique column type. `dbo:Group` has been assigned to rows have Rank1, 2, 4, and 9. `dbo:Person` has been assigned to remained rows. In this situation, we could not determine a unique column type.

References

1. Lehmberg, Oliver, et al. "A large public corpus of web tables containing time and context metadata." Proceedings of the 25th International Conference Companion on World Wide Web. International World Wide Web Conferences Steering Committee, 2016.
2. Kruit, Benno, Peter Boncz, and Jacopo Urbani. "Extracting Novel Facts from Tables for Knowledge Graph Completion." International Semantic Web Conference. Springer, Cham, 2019.
3. Naseer, Aisha, et al. "LOD for all: Unlocking infinite opportunities." Semantic Web Challenge (2014).
4. Semantic Web Challenge on Tabular Data to Knowledge Graph Matching `http://www.cs.ox.ac.uk/isg/challenges/sem-tab/`. (accessed Sep. 2019)
5. Efthymiou, Vasilis, et al. "Matching web tables with knowledge base entities: from entity lookups to entity embeddings." International Semantic Web Conference. Springer, Cham, 2017.
6. Zwicklbauer, Stefan, et al. "Towards Disambiguating Web Tables." International Semantic Web Conference (Posters & Demos). 2013.
7. Robertson, Stephen, and Hugo Zaragoza. "The probabilistic relevance framework: BM25 and beyond." Foundations and Trends in Information Retrieval 3.4 (2009): 333-389.
8. Okazaki, Naoaki, and Jun'ichi Tsujii. "Simple and efficient algorithm for approximate dictionary matching." Proceedings of the 23rd International Conference on Computational Linguistics. Association for Computational Linguistics, 2010.
9. Fang, Lu, Qingliang Miao, and Yao Meng. "DBpedia Entity Type Inference Using Categories." International Semantic Web Conference (Posters & Demos). 2016.