GBMTab: A Graph-Based Method for Interpreting Noisy Semantic Table to Knowledge Graph

Lianzheng Yang[1], Shuyang Shen[2], Jingyi Ding[3], and Jiahui Jin[3]✉

[1]School of Cyber Science and Engineering, Southeast University, China
[2]Chien-Shiung Wu College, Southeast University, China
[3]School of Computer Science and Engineering, Southeast University, China
{lianzhengyang,shua,jingyi_ding,jjin}@seu.edu.cn

Abstract. Tabular data on the web contains rich semantic information. However, it is challenging to match noisy tabular data to knowledge graphs. In this paper, we propose a framework called GBMTab. GBMTab is a semantic table interpretation framework that uses a robust probabilistic graphical model (PGM) method to solve entity linking and column type annotation problems with tabular data. Through this framework we participated in two tasks of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2021), i.e., CEA (Cell Entity Annotation), CTA (Column Type Annotation). The results of SemTab 2021 challenge show that our framework has positive performance.

Keywords: Tabular data · Entity linking · Table interpretation · GBMTab.

1 Introduction

Tables on web pages have become a high-quality source for applications such as knowledge extraction [1] and knowledge graph augmentation [2], as they contain a large amount of relational data covering information in many fields. Tabular data on the web contains rich semantic information, these data and corresponding semantic information are also recorded in some specific knowledge graphs (KGs), so match the tabular data into knowledge graph has critical research value.

In recent years, there have been lots of works on semantic annotation with tabular data, many of which use probabilistic graphical models to annotate tabular data. Limaye et al. [3] and Mulwad et al. [4] respectively proposed a probabilistic graphical model that captures semantic information in tables and annotates cells, columns and column pairs as entities, types and relationships in KGs. According to the results of SemTab 2020, the best performing approach is

MTab4Wikidata [5], which improves the results of entity search by using fuzzy search and statement search to deal with noise mentions.

However, it is arduous for machines to interpret semantic tabular data because of the diversity of languages and noise mentions. The Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab 2021) aims at benchmarking tabular data to knowledge graph matching systems. There are three tasks: Column Type Annotation (CTA), Cell Entity Annotation (CEA) and Column Property Annotation (CPA). The CTA task is assigning a semantic type to a column, the CEA task is matching cells to entities in a specific KG, and the CPA task is assigning a KG property to the relationship between two columns. These three tasks and their formal definitions can be illustrated by Figure 1.

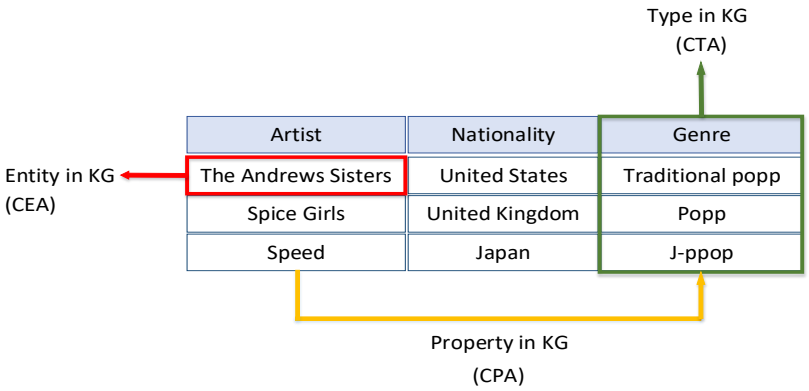

Fig. 1. Three sub-tasks of SemTab 2021.

We have proposed an approach to solve the CTA and CEA tasks, where the result of CEA task is the input of CTA task. For the CEA task, we build a graphical model and used an iterative probability propagation algorithm to find the referent entity for a cell. For the CTA task, we matched types of columns by using Wikidata SPARQL endpoint and feature engineering.

The remainder of this paper is organized as follows. In Section 2 we introduce the approach we proposed to solve the CEA and CTA tasks. Then, we describe the results for two rounds in Section 3 and make a conclusion in Section 4.

2 Methodology

The framework flowchart of the CEA and CTA tasks is showed in Figure 2, where the CEA task is composed of candidate entity generation and entity disambiguation, and the CTA task is composed of relation match and disambiguation. An

input form first passes the CEA task, it should be noted that in this step we introduce the noise mention repair mechanism. After the CEA task is completed, the marked form is used as the input of the CTA task, annotating the column type according to the amount of available information of the corresponding mention in the table. The following part is the detail of our specific approach.

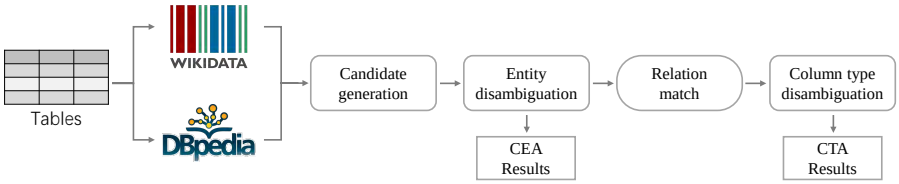

Fig. 2. The framework flowchart of the CEA and CTA tasks.

2.1 Entity linking

The entity linking is divided into two parts: candidate generation and entity disambiguation. For round 1, we select the challenge which link tabular data to the knowledge graph of DBpedia 2016-10. In round 2, we linked "BioTable" to the knowledge graph of Wikidata.

2.1.1 Candidate generation

For each mention in tabular data, we first need to find the corresponding candidate entity in the given knowledge graph. Due to the different composition of the knowledge graph data, we classified the different knowledge graphs in the candidate entity generation part. It is worth mentioning that in this step we have added a noise mention repair mechanism.

As for DBpedia, we use the following methods.

String similarity comparison: We build an index with a hash table for the DBpedia ontology and the corresponding link by using DBpedia 2016-10 datasets. Then we calculate similarity traversal between all DBpedia ontologies and mentions to determine which entity is the candidate for the mention. We define s as a mention in tabular data, e is an entity in the knowledge graph. Thereupon, we define the string similarity $StringSimilarity(s, e)$ as follows:

$$StringSimilarity(s, e) = 1 - \frac{LevenshteinDistance(s, e))}{sum\{length(s), length(e)\}} \tag{1}$$

We set a threshold to determine whether to add an entity e to the candidate set. If the threshold is set overly high, ground truth entity may not be included. On the contrary, if the threshold is set excessively low, it will increase the difficulty of subsequent task for entity disambiguation. Through experiments show in Figure 3, we randomly selected 10,000 mentions to generate candidate sets

under different threshold settings. It turns out that setting a threshold of 0.7 is a good choice.

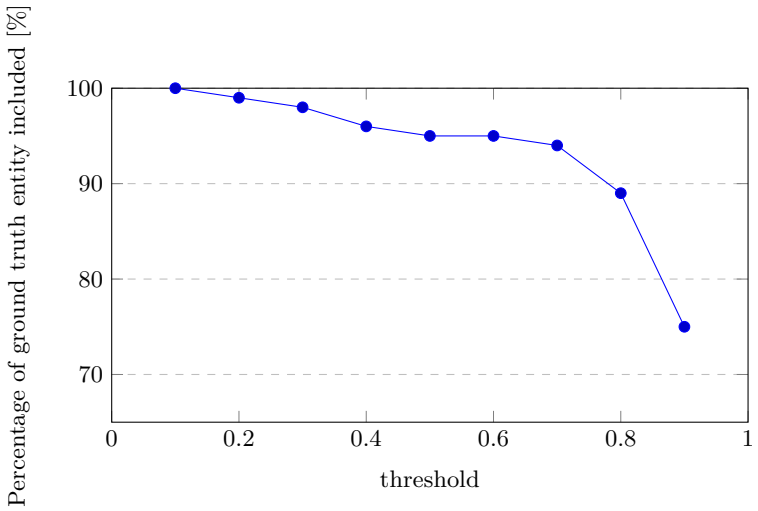

Fig. 3. The influence of the threshold on candidate set.

Noise mentions repair: After adding noise for some short length mentions, for example "cat" becomes "catt". Due to the short length, the string similarity will be lower and thus will be ignored. This is very fatal for the subsequent task of entity disambiguation. In order to solve noise mentions problem, we use search engine to solve this problem. Through Google search engine, it may automatically correct keywords containing the noise mentions and continue searching according to the recommended keywords. If search results have an impact on the judgment, we will also limit search scope to the domain name of DBpedia.

Multilingual: For mentions whose language is not English, we have introduced multilingual DBpedia datasets.

As for Wikidata, we use the following methods.

Wikidata MediaWiki API: Unlike DBpedia, in Wikidata, we query MediaWiki API by posting the mention and setting the limits to a maximum of 50. Wikidata will perform fuzzy search and internal sorting. We can think that the higher-ranking entities are more related to mentions.

Correction of noise mentions: This step is the same as that of DBpedia, we use the Google search engine.

2.1.2 Entity disambiguation

https://www.mediawiki.org/wiki/API:Search

In this step, our goal is to select an entity that best meets the contextual semantics from the candidate set. We assume that in the table, mentions in the same row or column are related. Thus, we set up such a model, which uses a fully connected network to build a graphical model, and uses the iterative probability propagation algorithm [6] to find a set of entities that are most likely to express semantics, jointly disambiguate mentions in table. These steps can be illustrated by Figure 4.

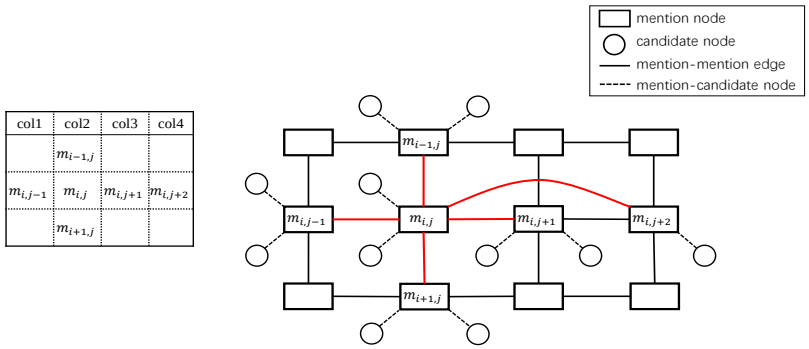

Fig. 4. The flowchart of entity disambiguation.

i) **Build disambiguation graph**: Starting from a given mention, create a disambiguation graph of the other mentions and the corresponding candidates which in the same row or column. For the mention m_{ij} in i^{th} row j^{th} column, add the i^{th} row mention $m_{i?}$ and j^{th} column mention $m_{?j}$ into the graph as mention node, then add candidate sets $S(m)$ corresponding to these mentions into the graph as candidate nodes.

ii) **Build features between nodes**: We use feature engineering to score the relationships between nodes to measure the semantic connection of nodes. The features consist of the following parts:
Priori Features: We calculate priori features from knowledge graph and WDC. For example, "New York" in Wikidata can be interpreted as New York City, New York State, Film and television, Literature, Music, etc. Thus, when entities link the word "New York", we tend to think of New York City or New York State, which appears more frequently. For DBpedia, we calculate the popularity of candidate entities, and for Wikidata, we take advantage of the order in which the MediaWiki API returns results.
In addition, we also consider the prior knowledge of context instead of individual mention itself, WDC is web table corpora which has extracted a large amount of web table data on the internet. For example, when "New

http://webdatacommons.org/webtables/

York" appeared on the web table with "Lou Reed", because the New York album and its author appeared more frequently in WDC, we can consider "New York" in this table as the album.
Context Features: Tabular structure is naturally composed of an ordered set of rows and columns [7]. For the cell c_{ij} in i^{th} row j^{th} column. Commonly one row contains a key object and its properties. Thus, mention texts in the same row could be recognized as the extended information of the key object. Despite the subtle connections among cells in the same column, there are strong semantic similarities among them. Under some circumstances, key object could appear repeatedly, which usually indicates the repeat of the same row. This will help to improve the accuracy when noise occurs. For the reasons above, we take the values of cells in the same row or column of the objective cell as its features, using *Levenshtein distance* and *cosine distance* to rank candidate entities.
Abstract Features: In round 1 of SemTab 2021, tasks can be fulfilled with DBpedia data. Compared to Wikidata, DBpedia has less data, but the data is more structured. Many entities have their own abstracts which describe the general information of entity. We can easily access the abstract of an entity with local DBpedia SPARQL endpoint. The search can make an intersection with the other available text features and score the corresponding entity with *cosine distance* as long as we have it appropriately tokenized.

iii) **Iterative probability propagation**: Through iterative probability propagation, based on the current values of nodes, greedily iterative assigns its maximum likelihood value. Then constantly calculates and updates features of the mention, and finally reaches the global optimal solution. Here, we formally define the state of the graphical model under initialization as P_0, the maximum number of iterations is T, the current number of iterations is t, when $t == T$ or $P_t == P_{t-1}$ probability propagation is finish, the P_t is output as a global disambiguation result.

2.2 Column type annotation

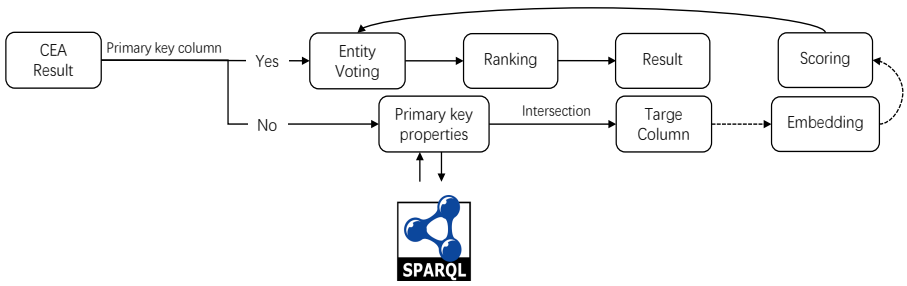

Fig. 5. The flowchart of column type annotation.

The CTA task can be performed by exploiting the process described in Figure 5. We use the result of the CEA to do the task. The first step is to identify the primary key column of the targets. Then adapted the relation match (Intersection in Figure 5) and embedding similarity to rank the candidates which are searched with the SPARQL endpoints.

2.2.1 Relation match

After annotating cell values, we search different types of each of these entities in the same column by using Wikidata SPARQL endpoint. Since there might be a lot of wrong candidates in the search results. It is essential to clean candidate set of each cell in the table to apply further mechanisms like voting and embedding to disambiguation.

Here we assume the mentions in primary key columns are all appropriately linked to corresponding entities while mentions not in primary key columns are commonly linked to a "vogue entity".

For example, Trump and American are texts of two cells in the same row, Trump can be easily linked to "Donald Trump" while American maybe linked to "The United States of America", which could not precisely describe semantics within the column: birth place of Trump. When we search for the type of "The United States of America" the results shall include "Country", "Superpower", etc., which do not involve the relation "place of birth". Thus, we use knowledge graph structure, trying to choose the most suitable relation path [3] for those entities whose values of properties exactly match the "vogue entity". And thus extract the type we need in ontology. Then we use vote mechanism, embedding distance [8] [9] and text similarity to rank types in the candidate set and finally get result.

2.2.2 Column type disambiguation

If the intersection of properties and search candidates is empty, we need to use feature engineering to find the most suitable linking entity instead of simple voting. With word embedding, we can achieve better accuracy when it comes to correlation in semantic level.

Considering word embedding to extend available information within a table for the CTA task is reliable. Since the mention text on the CEA task is hard to apply word embedding due to the uncommon words which are out of vocabulary, the trending method on CTA is voting [10], ignoring properties of the cell in primary key column. However, if the intersection of properties and search candidates is empty, we use a hybrid method which consists of vote mechanism, embedding distance and text similarity to rank types in candidate set.

3 Results

Table 1 shows the results of our first two rounds of challenge in SemTab 2021.

Table 1. The SemTab 2021 results of our team.

	Round1		Round2	
	F1	Precision	F1	Precision
CEA	0.692	0.692	0.868	0.868
CTA	0.133	0.133	-	-

3.1 Result for the CEA task

For the CEA task in round1, we use DBpedia 2016-10 data as the knowledge graph. It can be seen in Table 2 that the addition of noise to the table has an impact on the candidate entities generated based on string similarity. After adding the noise mentions repair mechanism, we can see that F1 and precision have been greatly improved which proves that this method is effective. In round2, we used Wikidata as our knowledge graph. In this round of challenge, we focused on the impact of candidate entity generation on the CEA results. After experimentation, we are more inclined to use MediaWiki API, which can perform fuzzy searches on characters. It is obvious that the precision is better than round1. Figure 6 shows that by experimenting with different range of top-k on the candidate set which contains ground truth entity, we found that our probabilistic graphical model has strong robustness.

Table 2. The impact of noise mentions repair in round1.

	F1	Precision
Without Repair	0.502	0.502
Repair	0.692	0.692

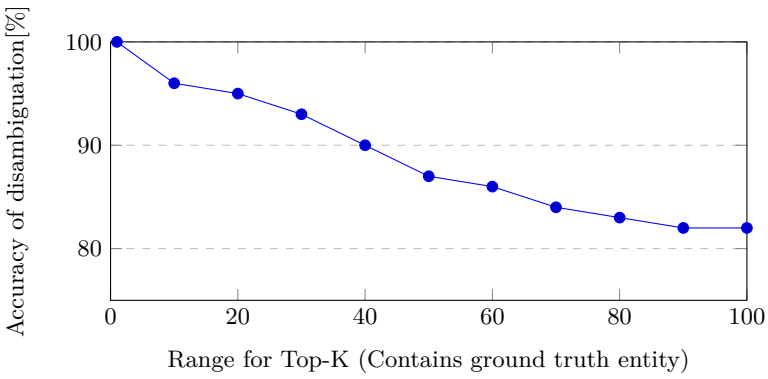

Fig. 6. The impact of the range for Top-K on PGM model.

3.2 Result for the CTA task

For the CTA task, the introduction of encoding model and elements in primary key column appears to regularize candidate list at the level of semantics and give less weight to the coarse-grained candidates. Even though the intersection of properties of key value and target mention mentioned in Section 2.2, we still remove the most semantically irrelevant entities to guarantee the candidate list is as clean as possible. A better solution might be retraining pre-trained BERT to fine-tune BERT model. Since the language that ontology uses is restricted (English almost) and the word tokens applied in ontology are also high-frequency words included in vocabulary. This could be future work of the task.

4 Conclusion

Graphical model of iterative probability propagation has obvious effect in entity disambiguation, and candidate generation as its upstream task has a greater impact with disambiguation results. Spell correction and noise mentions detection in the CEA task can improve the performance for the CTA task. Also, the size of the table has a great influence on the speed, it will take a lot of time to propagate the probability of graph with large table. In addition, the application of BERT embedding and property intersection helped to improve the result for the CTA task. In comparison with the dataset in round 1, the dataset of the following rounds is more complicated and field-specific. DBpedia could extend the information of some entities while helps little to the uncommon and field-specific entities.

In the future, we will pay more attention to accuracy and speed of candidate entity generation and entity disambiguation, we will also aim at improving column type annotation by using pre-trained model to integrate the information, making the result more fine-grained.

5 Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grants No. 62072099, Jiangsu Provincial Key Laboratory of Network and Information Security under Grants No.BM2003201, Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grants No.93K-9, and the Fundamental Research Funds for the Central Universities.

References

1. Daheng Wang, Prashant Shiralkar, Colin Lockard, Binxuan Huang, Xin Luna Dong, and Meng Jiang. TCN: Table Convolutional Network for Web Table Interpretation. In proceedings of the Web Conference 2021, pp. 4020–4032, April 2021.

2. Dominique Ritze, Oliver Lehmberg, Yaser Oulabi, and Christian Bizer. Profiling the Potential of Web Tables for Augmenting Cross-domain Knowledge Bases. In proceedings of the 25th international conference on world wide web, pp. 251-261, April 2016.
3. G. Limaye, S. Sarawagi, and S. Chakrabarti, "Annotating and searching web tables using entities, types and relationships," Proc. VLDB Endow., vol. 3, pp. 1338–1347, September 2010.
4. Varish Mulwad, Tim Finin, and Anupam Joshi. Semantic message passing for generating linked data from tables. In: Alani, H., Kagal, L., Fokoue, A., Groth, P., Biemann, C., Parreira, J.X., Aroyo, L., Noy, N., Welty, C., Janowicz, K. (eds.) ISWC 2013, Part I. LNCS, vol. 8218, pp. 363–378. Springer, Heidelberg (2013) .https://doi.org/10.1007/978-3-642-41335-3_23
5. Phuc Nguyen, Ikuya Yamada, Natthawut Kertkeidkachorn, Ryutaro Ichise, and Hideaki Takeda. MTab4Wikidata at SemTab 2020: Tabular Data Annotation with Wikidata. In SemTab@ ISWC, pp. 86-95, 2020.
6. Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. TabEL: Entity Linking in Web Tables. In Proceedings of the 14th International Conference on The Semantic Web (ISWC'15). 425–441.
7. Yusra Ibrahim, Mirek Riedewald, and Gerhard Weikum. Making Sense of Entities and Quantities in Web Tables. In Proc. of CIKM '16. 1703–1712, 2016.
8. Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles A. Sutton. ColNet: Embedding the Semantics of Web Tables for Column Type Prediction. In AAAI, 2018.
9. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT.
10. S. Tyagi and E. Jimenez-Ruiz. LexMa: Tabular Data to Knowledge Graph Matching using ´ Lexical Techniques. In Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab). CEUR-WS.org, 2020.