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

Lianzheng Yang1, Shuyang Shen2, Jingyi Ding1, and Jiahui Jin3

1School of Cyber Science and Engineering, Southeast University, China
2Chon-Shan Wu College, Southeast University, China
3School of Computer Science and Engineering, Southeast University, China

jinn@seu.edu.cn

- **Background**: Tabular data on the Web contains rich semantic information, so matching the tabular data into knowledge bases is an important problem.
- **Challenge**: It is challenging to interpret semantic tabular data due to the diversity of languages and noise mentions.
- **Methods**: We proposed a semantic table interpretation framework called GBMTab to solve Cell Entity Annotation (CEA) and Column Type Annotation (CTA) tasks by using a probability graph model and a knowledge graph path-matching method.

### Introduction

<table>
<thead>
<tr>
<th>Entity in KB (CEA)</th>
<th>A type in KB (CTA)</th>
<th>Property in KB (CPA)</th>
</tr>
</thead>
</table>

Three sub-tasks of SemTab 2021.

### Method of CEA

- **Candidate generation**
  - **DBpedia**:
    - String similarity comparison: Define s as a mention and e as an entity.
    
    \[
    \text{StringSimilarity}(s, e) = 1 - \frac{\text{LevenshteinDistance}(s, e)}{\max(\text{length}(s), \text{length}(e))}
    \]
  - **Noise mentions repair**: Use Google search engine to correct noise mentions (e.g., “cat” becomes “catt”).
  - **Multilingual**: Introduce multilingual DBpedia datasets.

- **Wikidata**
  - Wikidata MediaWiki API: Query MediaWiki API by posting a mention and setting the limits to a maximum of 50.
  - **Correction of noise mentions**: Same as DBpedia.

- **Entity disambiguation**
  - **Build disambiguation graph**: Starting with a given mention (m_i), create a disambiguation graph of all other mentions in the same row or column and mention’s (m_j) corresponding candidates.
  - **Build features between nodes**:
    - **Priori Features**: Calculate priori features from Knowledge Base and WDC[1].
    - **Context Features**: Take the values of cells in the same row or column of objective cell as its feature, and use Levenshtein distance and cosine distance to rank the candidate entities.
    - **Abstract Features**: Intersect abstract of an entity with other available text features and score it with cosine distance.
  - **Iterative probability propagation**: Greedily assigns the current value of a node to its maximum likelihood value, continuously calculates and updates the feature of the mention, and finally reaches the global optimal solution.

### Method of CTA

- **Candidate generation**
  - DBpedia
  - String similarity comparison: Define s as a mention and e as an entity.
  - Noise mentions repair: Use Google search engine to correct noise mentions (e.g., “cat” becomes “catt”).
  - Multilingual: Introduce multilingual DBpedia datasets.

- **Wikidata**
  - Wikidata MediaWiki API: Query MediaWiki API by posting a mention and setting the limits to a maximum of 50.
  - **Correction of noise mentions**: Same as DBpedia.

- **Entity disambiguation**
  - **Build disambiguation graph**: Starting with a given mention (m_i), create a disambiguation graph of all other mentions in the same row or column and mention’s (m_j) corresponding candidates.
  - **Build features between nodes**:
    - Priori Features: Calculate priori features from Knowledge Base and WDC[1].
    - Context Features: Take the values of cells in the same row or column of objective cell as its feature, and use Levenshtein distance and cosine distance to rank the candidate entities.
    - Abstract Features: Intersect abstract of an entity with other available text features and score it with cosine distance.
  - **Iterative probability propagation**: Greedily assigns the current value of a node to its maximum likelihood value, continuously calculates and updates the feature of the mention, and finally reaches the global optimal solution.

### Result

<table>
<thead>
<tr>
<th>Round1</th>
<th>Round2</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Precision</td>
</tr>
<tr>
<td>CEA 0.692</td>
<td>0.692</td>
</tr>
<tr>
<td>CTA 0.133</td>
<td>0.133</td>
</tr>
<tr>
<td>CPA</td>
<td>-</td>
</tr>
</tbody>
</table>

**The SemTab 2021 results of our team**

- **Result for CEA**
  - Without Repair: 0.502
  - Repair: 0.692

- **Result for CTA**
  - The introduction of encoding models and elements in primary key column appears to regularize candidate list at the semantic level and give less weight to the coarse-grained candidates.

### Conclusion

- The iterative probability propagation graph model has obvious effects in entity disambiguation. The candidate generation as its upstream task has a greater impact on the disambiguation result. Spelling correction and noise detection in the CEA task can improve the performance of the CTA task.
- The size of table has a great influence on processing speed.
- The application of BERT embedding and property intersection helps to improve the results of CTA task.

### References

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