Disambiguating Web Tables using Partial Data

Ziqi Zhang

Department of Computer Science, University of Sheffield, UK
z.zhang@dcs.shef.ac.uk

Abstract. This work addresses disambiguating Web tables - annotating content cells with named entities and table columns with semantic type information. Contrary to state-of-the-art that builds features based on the entire table content, this work uses a method that starts by annotating table columns using automatically selected *partial* data (i.e., a sample), then using the type information to guide content cell disambiguation. Different sample selection methods are introduced and tested to show that they contribute to higher accuracy in cell disambiguation, comparable accuracy in column type annotation with reduced computation.

1 Introduction

Enabling machines to effectively and efficiently access the increasing amount of tabular data on the Web remains a major challenge to the Semantic Web, as the classic indexing, search and NLP techniques fail to address the underlying semantics carried by tabular structures [1, 2]. This has sparked increasing interest in research on semantic Table Interpretation, which deals with semantically annotating tabular data such as shown in Figure 1. This work focuses specifically on annotating table columns that contain named entity mentions with semantic type information (column classification), and linking content cells in these columns with named entities from knowledge bases (cell disambiguation). Existing work follows a typical workflow involving 1) retrieving candidates (e.g., named entities, concepts) from the knowledge base, 2) constructing features of candidates, and 3) applying inference to choose the best candidates. One key limitation is that they adopt an *exhaustive* strategy to build the candidate space for inference. In particular, annotating table columns depends on candidate entities from *all* cells in the column [1, 2]. However, for human cognition this is unnecessary. For example, one does not need to read the entire table shown in Figure 1 - which may contain over a hundred rows - to label the three columns. Being able to make such inference using *partial* (as opposed to the entire table) or *sample* data can improve the efficiency of the task as the first two phases can cost up to 99% of computation time [1].

Sample driven Table Interpretation opens up several challenges. The first is defining a sample with respect to each task. The second is determining the optimal size of the sample with respect to varying sizes of tables. The third is choosing the optimal sample entries, since a skewed sample may damage accuracy. Our previous work in [5] has proposed TableMiner to address the first two challenges. This work adapts TableMiner to explore the third challenge. A number of sample selection techniques are introduced and experiments show that they can further improve cell disambiguation accuracy and in the column type annotation task, contribute to reduction in computation with comparable learning accuracy.

2 Related Work

An increasing number of work has been carried out in semantic Table Interpretation, such as Venetis et al. [3] that uses a maximum likelihood model, Limaye et al. [1] that uses a joint inference model, and Mulwad et al. [2] that uses joint inference with semantic message passing. These methods differ in terms of the inference models, features and background knowledge bases used. All these methods are, as discussed earlier, 'exhaustive' as they require features built based on all content cells in order to annotate table columns. Zwicklbauer et al. [6] is the first method that annotates a table column using a sample of the column. However, the sample is arbitrarily chosen.

NE-column

Name	Area	Prefecture	
Trichonida	96,513	Aetolia-Acarnania	Content cell
Yliki	22,731	Boeotia	
Amvrakia	13,619	Aetolia-Acarnania	
Lysimachia	13,200	Aetolia-Acarnania	
…	…	…	

Fig. 1. Lakes in Central Greece

3 Methodology

TableMiner is previously described in [5]. It disambiguates named entity columns in a table in two phases. The *first phase* creates preliminary annotations by using a sample of a column to classify the column in an iterative, incremental algorithm shown in Algorithm 1. In each iteration, a content cell $T_{i,j}$ drawn from a column T_j is disambiguated (output $E_{i,j}$). Then the concepts associated with the winning entity are gathered to create a set of candidate concepts for the column, C_j. Candidate concepts are scored and their score can change at each iteration due to newly disambiguated content cells adding re-enforcing evidence. At the end of each iteration, C_j from the current iteration is compared with the previous. If scores of candidate concepts are little changed (convergence, see [5] for a method for detection), then column classification is considered to be stable and the highest scoring candidates are (C_j^+) chosen to annotate the column. The *second phase* begins by disambiguating the remaining cells (part I), this time using the type information for the column to limit candidate entity space to those belonging to the type only. This may revise C_j for the column, either adding new elements, or resetting scores of existing ones and possibly causing the winning concept for the column to change. In this case, the next part of the second phase (part II) repeats the disambiguation and classification operations on the entire column, while using the new C_j^+ as constraints to restrict candidate entity space. This procedure repeats until C_j^+ and the winning entity in each cell stabilizes (i.e., no change).

Modified TableMiner For the purpose of this study, TableMiner is modified (TM_{mod}) to contain only the *first phase* and *part I* of the *second phase*. In other words, we do not revise the column classification results obtained from sample data. Therefore TM_{mod} may only use a fraction of a column's data to classify the column, which reduces computation overhead compared to classic 'exhaustive' methods.

Sample selection The choice of the sample can affect learning in TM_{mod} in two ways. While the *size* of the sample is dealt with by the convergence measure described in [5], here we address the issue of selecting the *suitable* sample entries to ensure learning accuracy. Since column classification depends on the disambiguated cells in the sample, we hypothesize that high accuracy of cell disambiguation contributes to high accuracy in column classification. And we further hypothesize that higher accuracy of content cell disambiguation can be achieved by 1) richer feature representation, and 2) less ambiguous names (i.e., if a name is used by only one or very few named entities). Therefore, we propose three methods to compute a score of each content cell in a column, then rank them by the score before running Algorithm 1 (i.e., input T_j will contain content cells the order of which is re-arranged based on the scores).

Algorithm 1 Sample based classification

```
Input: T_j; C_j ← ∅
for all cell T_{i,j} in T_j do
    prevC_j ← C_j
    E_{i,j} ←disambiguate(T_{i,j})
    C_j ←updateclass(C_j, E_{i,j})
    if convergence(C_j, prevC_j) then
        break
    end if
end for
```

One-sense-per-discourse (ospd) First and foremost, we make the hypothesis of 'one-sense-per-discourse' in table context, that if an NE-column is not the subject column of the table (e.g., the first column in Figure 1 is a subject column), then cells containing the same text content are extremely likely to express the same meaning[1]. Thus to apply *ospd* we firstly re-arrange cells in a column by putting those containing duplicate text content adjacent to each other. Next, when disambiguating a content cell, the feature representation of the cell concatenates the row context of the cell, and that of any adjacent cells with the same text content (e.g., in Table 1 we assume 'Aetolia-Acarnania' on the three rows to have the same meaning, and build a single feature representation by concatenating all the three rows). Effectively this creates a richer feature representation for cells whose content re-occur across a table.

Feature size (fs) With *fs*, we firstly apply *ospd*, then rank cells in a column by the size of their feature representation as determined by the number of tokens in a bag-of-words representation. This would allow TM_{mod} to start with cells that potentially have the largest - hence 'richest' - feature representation in Algorithm 1.

Name length (nl) With *nl*, we count the number of words in the cell text content to be disambiguated and rank cells by this number - name length (e.g., in Table 1 'Aetolia-Acarnania' has two words and will be disambiguated before 'Boeotia'). *nl* merely relies on the name length of a cell content and does not apply *ospd*. The idea is that longer names are less likely to be ambiguous.

4 Evaluation and Conclusion

We evaluate the proposed methods of sample selection using two datasets: LimayeAll and Limaye200[2]. LimayeAll contains over 6000 tables and is used for evaluating content cell disambiguation. Limaye200 contains a subset 200 tables from LimayeAll with

[1] Due to space limitation, details are omitted but can be found in [3, 4]

[2] [4], currently under transparent review.

Cell disambiguation (LimayeAll)				**Column classification** (Limaye200)			
TM_{mod}	TM^{ospd}_{mod}	TM^{fs}_{mod}	TM^{nl}_{mod}	TM_{mod}	TM^{ospd}_{mod}	TM^{fs}_{mod}	TM^{nl}_{mod}
0.809	0.812	0.812	0.813	0.723	0.719	0.721	0.723

Table 1. Cell disambiguation and column classification accuracy in F1.

columns manually annotated with Freebase concepts, and used for evaluating column classification. As a **baseline**, TM_{mod} without any sample selection techniques is used. It simply chooses cells from a column in their original order in Algorithm 1. This is compared against TM^{ospd}_{mod}, which applies $ospd$ to non-subject NE-columns, preserves the original order but disambiguates groups of cells containing the same text content; TM^{fs}_{mod} that applies $ospd$ to non-subject NE-columns then prioritizes cells that potentially have richer feature representation; and TM^{nl}_{mod} that prioritizes cells containing longer text content. Results on both datasets are shown in Table 1. It suggests that, compared against TM_{mod}, the sample selection techniques can enhance the accuracy of cell disambiguation marginally. In the column classification task however, they do not add benefits in terms of accuracy. By analyzing the computation overhead in terms of the automatically determined sample size in each table, it shows that the sample selection techniques have reducing the amount of data to be processed in column classification. As an example, TM_{mod} converges on average after processing 58% of cells in a table column, i.e., it manages to classify a table column using a sample size of 58% of the total number of cells in that column. TM^{ospd}_{mod} reduces this to 53%, for TM^{fs}_{mod} 52% and for TM^{nl}_{mod} 58% (unchanged). This may contribute to noticeable reduction in CPU time since the construction of feature space (including querying knowledge bases) for each data unit consumes over 90% of computation time [1]. To summarize, it has been shown that, by using sample selection techniques, it is possible to semantically annotate Web tables in a more efficient way, achieving comparable or even higher learning accuracy depending on tasks.

Acknowledgement: Part of this work is carried out in the LODIE project (Linked Open Data Information Extraction), funded by EPSRC (EP/J019488/1).

References

1. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and searching web tables using entities, types and relationships. Proceedings of the VLDB Endowment 3(1-2), 1338–1347 (2010)
2. Mulwad, V., Finin, T., Joshi, A.: Semantic message passing for generating linked data from tables. In: International Semantic Web Conference (1). pp. 363–378. Springer (2013)
3. Venetis, P., Halevy, A., Madhavan, J., Paşca, M., Shen, W., Wu, F., Miao, G., Wu, C.: Recovering semantics of tables on the web. Proc. of VLDB Endowment 4(9), 528–538 (Jun 2011)
4. Zhang, Z.: Start small, build complete: Effective and efficient semantic table interpretation using tableminer. In: The Semantic Web Journal (under reviewer, #668-1878) (2014)
5. Zhang, Z.: Towards efficient and effective semantic table interpretation. In: To appear in: ISWC2014 (2014)
6. Zwicklbauer, S., Einsiedler, C., Granitzer, M., Seifert, C.: Towards disambiguating web tables. In: International Semantic Web Conference (Posters & Demos). pp. 205–208 (2013)