Towards a Large Corpus of Richly Annotated Web Tables for Knowledge Base Population

Basil Ell*, Sherzod Hakimov*, Philipp Braukmann, Lorenzo Cazzoli, Fabian Kaupmann, Amerigo Mancino, Junaid Altaf Memon, Kai Rother, Abhishek Saini, and Philipp Cimiano

CIT-EC, Universität Bielefeld
{bell, shakimov} @cit-ec.uni-bielefeld.de

Abstract. Web Table Understanding in the context of Knowledge Base Population and the Semantic Web is the task of i) linking the content of tables retrieved from the Web to an RDF knowledge base, ii) of building hypotheses about the tables' structures and contents, iii) of extracting novel information from these tables, and iv) of adding this new information to a knowledge base. Knowledge Base Population has gained more and more interest in the last years due to the increased demand in large knowledge graphs which became relevant for Artificial Intelligence applications such as Question Answering and Semantic Search.
In this paper we describe a set of basic tasks which are relevant for Web Table Understanding in the mentioned context. These tasks incrementally enrich a table with hypotheses about the table's content. In doing so, in the case of multiple interpretations, selecting one interpretation and thus deciding against other interpretations is avoided as much as possible. By postponing these decision, we enable table understanding approaches to decide by themselves, thus increasing the usability of the annotated table data.
We present statistics from analyzing and annotating 1.000.000 tables from the Web Table Corpus 2015 and make this dataset as well as our code available online.[1]

Keywords: Information Extraction, Table Interpretation, Corpus Creation, Corpus Annotation, Hypothesis Creation

1 Introduction

Large amounts of information are available on the Web. However, most information is not processable by machines in a way which would allow machines to perform semantic search on this content or to answer questions using this data. Having data represented in the RDF (Resource Description Format) format would be one possibility towards this goal. Despite the progress made in

* Corresponding author

[1] The data is available at `http://doi.org/10.4119/unibi/2912802`, the code is available at `https://github.com/isywtu/code`, and the website is located at `https://isywtu.github.io/website`.

the field of Natural Language Understanding, extracting information from textual documents and representing the content in RDF remains limited due to the complexity of natural language.

Besides natural language texts, the Web also contains a plethora of tables and it might be easier to extract information from tables due to their inherent structure (e.g., rows in a table may be similar to each other) than from text.

An example of a table is shown in Table 1. A human possessing general knowledge could assume that this table is about American politicians, their dates of birth, and the parties they belong to. That means, humans can use their knowledge to build hypotheses about the data. Having hypotheses about some parts of the table enables them to obtain new information from other parts.

Given that more and more RDF data became available online, e.g., in the form of knowledge bases such as DBpedia and as Linked Open Data in general, we have machine-processable information available so that machines can build hypotheses about the content of tables, develop an understanding of the schema underlying a table, and add extracted data to knowledge bases. RDF data is thus applied as a leverage for Information Extraction from Web tables.

In this paper we present basic tasks that enrich a large subset of tables of an existing dataset – the WDC Web Table Corpus 2015 (WTC) – with hypotheses based on DBpedia. Tables enriched with hypotheses created by basic tasks – such as table normalization or entity linking – allow others to focus on higher-level tasks of table understanding, such as column understanding, and to investigate how much information extracted from Web tables could be added to DBpedia.

Our main contributions are: i) we present eight basic tasks that create hypotheses about (parts of) tables. ii) We enrich a corpus of 1,000,000 tables with hypotheses, thus allowing other researchers to focus on higher-level tasks. An important aspect is that in the case of multiple possibly disagreeing interpretations, selecting one interpretation and thus deciding against other interpretations is avoided. By postponing these decisions we enable table understanding approaches to decide by themselves, thus increasing the usability of the annotated table data. iii) We present statistics about 1 million tables and the results of applying our tasks on these tables, and iv) we make all annotated data, the code as well as further statistics and example tables available.

The remainder of this paper are structured as follows: Section 2 presents the basic tasks, how they build on the WTC data, and how hypotheses of one tasks are built on top of hypotheses created by other tasks. Section 3 presents statistics about the analyzed data, Section 4 discusses related work, and Section 5 concludes the paper.

2 Basic Table Interpretation Tasks

In this section we describe basic table interpretation tasks. We refer to them as *basic*, since they create hypotheses which are prerequisites to understanding the entire table. For example, a basic hypothesis concerns whether a cell is a header cell or a data cell (see *table segmentation*, Section 2.4).

Table 1. Example of a horizontal table (adapted example from the paper *Understanding Tables on the Web* by Wang et al. [18]).

Politician	Surname	Date of Birth	State	Political Party
Barack Obama	Obama	Aug 4, 1961	Illinois	-
George W. Bush	Bush	July 6, 1946	Texas	Republican
Hillary Clinton	Clinton	Oct 26, 1947	-	Democratic

We created our framework for Web table understanding for the *WDC Web Table Corpus 2015*[2] which consists of 233 million tables in JSON format. This corpus was created from a set of 1.78 billion HTML pages from the *Common Crawl July 2015 Web corpus*.[3] At the time that corpus was created, several decisions were made by the authors of that corpus: tables that contain tables in their cells as well as small tables (those consisting of less than two columns or three rows) were excluded. The set of exclusion criteria contains further constraints which we do not discuss here. Besides the exclusion of tables, four other aspects were decided and are annotated in the corpus. We take these annotations as truth and do not reimplement these tasks but we transform these annotations into hypotheses, so that our tasks can build hypotheses on top of these hypotheses. We refer to these existing tasks as WTC (Web Table Corpora) tasks.

Hypotheses are represented in JSON format and are organized according to what part of a table a hypothesis is about: table, row, column, and cell. For example, the orientation (horizontal or vertical) of a table is a hypothesis about the table, whereas an entity mentioned in a cell is a hypothesis about a cell.

Hypothesis generation is an incremental process: tasks can be executed repeatedly, thereby adding hypotheses to a table, and hypotheses can be related to hypotheses added in previous steps. For example, *language detection* (see Section 2.6) might assume a text to be in German. Based on this hypothesis, given a cell value `10.11.12`, *table normalization* (Section 2.7) then assumes the string to be a date in day-month-year format (day=10, month=11, year=2012), whereas it would assume the string to be a date in month-day-year format (day=11, month=10, year=2012) in case where the detected language is American English. Note however, that we allow contradictory hypotheses. In the example above it could be the case that *language detection* identifies both American English and German and thus creates two language hypotheses. Then, for each language hypothesis the *table normalization* task creates another hypothesis. As another example of the incremental process, a table might initially be not excluded from the corpus. However, after executing *entity linking* (Section 2.8), the table might be excluded if too few entities are identified.

The hierarchy of the tasks is shown in Figure 1. Note that some tasks process data from the WTC corpus directly (e.g., *language detection*), other tasks only process hypotheses created by other tasks (e.g., *entity linking* processes hypothe-

[2] This corpus is available at `http://webdatacommons.org/webtables/`

[3] `http://commoncrawl.org/2015/08/july-2015-crawl-archive-available/`

ses created by *table normalization*). So far, not all hypotheses are processed by some task, but may be processed by some non-basic tasks is the future.

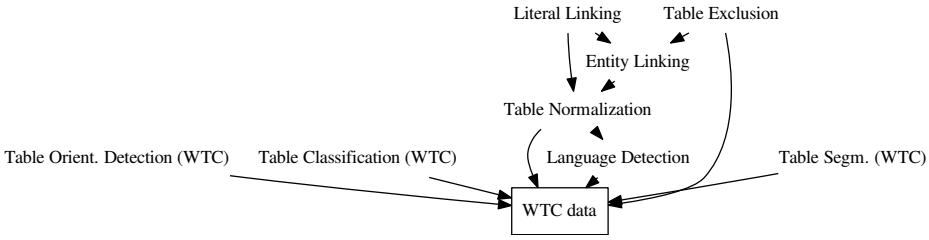

Fig. 1. Hierarchy of basic tasks and the data they process. For example, *literal linking* processes hypotheses created by *entity linking* and *table normalization*, whereas *table orientation detection* processes data from the WTC corpus directly.

2.1 Scheduling Task

Scheduling is the task of deciding which tasks to execute on a table and in which order to execute the tasks. The scheduling task reads in a table, sends it to one of the other tasks and retrieves the annotated table. Our scheduler sends the table data to the tasks in the following order: *language detection*, *table normalization*, *entity linking*, and *literal linking*. Note that this is a simple approach and one could also introduce more complex cyclic orders. After each of these steps, the table is sent to the *table exclusion* task. If a table should be excluded, no further tasks are called upon. Otherwise, the scheduler proceeds in the above order. After all tasks have been applied to a table or the table has been excluded, the scheduler stores the table.

2.2 Table Orientation Detection Task (WTC task)

This WTC task identifies the orientation of a table, where the orientation can either be *horizontal* (i.e., for some columns it is the case that they stand for an attribute) or *vertical*. We translate the WTC annotation into a table-related hypothesis of type `table orientation` (thus we do not reimplement their approach), for example as shown in Figure 2 for the horizontal table in Table 1.

Fig. 2. H0 – A table orientation hypothesis related to the example table (Table 1).

```
"H0" : {
 "created_by_task": "table orientation detection",
 "hypothesis_type": "table orientation",
 "orientation" : "horizontal",
 "source" : "WTC",
}
```

2.3 Table Classification Task (WTC task)

Table Classification is the task of classifying a table into one of the four classes *relational table*, *entity table*, *matrix table*, and *layout table*, as done by the creators of the WTC corpus and as described in [4] – thus we do not reimplement their approach. Examples of these table types can be found on the WTC website. The example table (Table 1) can be classified as *relational table*. We translate the WTC annotation into a table-related hypothesis of type `table classification`, for example as shown in Figure 3.

Fig. 3. H1 – A table classification hypothesis related to the example table (Table 1).

```
"H1" : {
  "created_by_task": "table classification",
  "hypothesis_type": "table classification",
  "classification" : "relational",
  "source" : "WTC",
}
```

2.4 Table Segmentation Task (WTC task)

Table Segmentation is the task of segmenting a table into header areas and data areas. Header row detection is already done for the WTC tables. In principle, the task could go beyond header row detection, since, for example, tables can have a more complex structure where the first column contains headers, too. We do not implement the segmentation task but rely on the WTC data. We create the following row-based or column-based hypotheses as shown in Figure 4. Whether this hypothesis is column-based or row-based depends on whether it is added to the list of column-based or row-based hypotheses.

Fig. 4. H2 – A table segmentation hypothesis related to the example table (Table 1).

```
"H2": {
   "created_by_task": "table_segmentation",
   "hypothesis_name": "table_segmentation",
   "header_row": [0],
   "source": "WTC"
}
```

2.5 Table Exclusion Task

The purpose of this task is to exclude tables from further processing if it seems like a table does not contain information that could be added to the knowledge base or if it is unlikely that valid information can be extracted from the table. For example, if the table seems to be used for layout purposes in a webpage only (e.g., for the menu of the page), or if no cell can be linked to an entity in our knowledge base, such as if a table only consists of numerical values, then the information probably does not fall into the domain of the knowledge base, or if the table structure is too complex (e.g., with multiple header rows and columns), then it is unlikely that we can arrive at a correct understanding.

For this task we rely on the table type classification of WTC. All tables that are not classified as *relational table* (see Section 2.3) are excluded. After executing the *entity linking* task (see Section 2.8) on a table, the table is excluded if no entity linking hypotheses were generated.

2.6 Language Detection Task

Language Detection is the task of detecting one or more languages for a given table. Therefore, raw cell data is analyzed and table-based language hypotheses with confidence values are created. Language information helps to reduce the complexity of finding the right information for the tasks of *table normalization*, *entity linking*, and *literal linking*. It facilitates finding correct formats for datatypes and unit measures. It also reduces computational costs for searching indices (i.e., those used by *entity linking* and *literal linking*).

We concatenate the content of all cells into a single string that is used as input to the language classification tool *langdetect*,[4] which computes the most probable languages with probabilities for 55 languages. An example hypothesis is shown in Figure 5. The language tags are ISO_639-1 tags.[5]

Fig. 5. H3 – A language hypothesis related to the example table (Table 1).

```
"H3" : {
   "created_by_task" : "language_detection", "lang" : "en",
   "confidence" : 0.9
}
```

2.7 Table Normalization Task

Table normalization is the task of normalizing values found in cells, such as transforming different representations of dates into a canonical representation so that other tasks do not need to take into account various representations.

Values, such as those representing weights, lengths, volumes, time etc. can have unit identifiers attached. For each string that appears to be a value followed by a unit identifier we create a hypothesis that contains both the value and the base unit identifier separately where the value is converted to the base unit (i.e., *kg* for weights). For example, given values of *10kg*, *100g*, *34t*, these are interpreted as weights and are converted to kilograms. A particular emphasis is given to the date representation because dates often occur in tables and multiple formats are possible (e.g., “4 August 1961”, “4-8-1961”, “Aug 1, 2016”, “August 1, 2016”, “1961/8/4” or “1961.8.4”). For each date that we detect, we create a hypothesis that contains the original value we found and also the `xsd:date` value. Note that hypotheses created by the *language detection* task are taken into account. This allows to test with regular expressions that are specific to a certain language, such as regular expressions for German date formats.

[4] https://pypi.python.org/pypi/langdetect
[5] https://www.iso.org/iso-639-language-codes.html

For each cell we always create a `plain` hypothesis that contains the original value and specifies the datatype *xsd:string*. The advantage of creating hypotheses about a cell's content is that other tasks do not have to look at the original data anymore but only need to scan for hypotheses. This can be seen in Figure 1 where the *entity linking* task only needs to process hypotheses created by the *table normalization* task. Another relevant aspect of the `plain` hypothesis: since our interpretation can go wrong, for example, a cell may contain the value `100` and it can be interpreted as a number or as a string, if we only create the hypothesis that this cell contains an integer value, then the *entity linking* task would ignore this cell, even though the table is about movies and `100` is actually a movie title. For each language we create another `plain` hypothesis.

Figure 6 shows an example of a hypothesis created for the string `Aug, 4, 1961` which was found in Table 1 in cell 2−1 (column-row). The types of hypotheses that we create are `date` as shown in Figure 6, `value` for integer or float values without unit identifiers, `value and unit` for values with unit identifier, and `plain`.

Fig. 6. A table normalization hypothesis related to the cell 2-1 (column-row) (with the content `Aug, 4, 1961`) in the example table (Table 1).

```
"H4": {
   "created_by_task": "table_normalization",
   "hypothesis_name": "date",
   "based_on_hypotheses": ["H3"]
   "data_type": "xsd:date",
   "original_value": "Aug, 4, 1961",
   "value": "1961-08-04",
   "year": 1961,
   "month": 8,
   "day": 4,
   "refers_to_row": 1,
   "refers_to_column": 2,
}
```

2.8 Entity Linking Task

This task links strings found in cells to DBpedia. For each type of resource (entity, property, class) and for each language we create another index of strings which are the names of the respective resources according to DBpedia. For properties and classes these are given via the property `rdfs:label`. For entities we use the same methodology as NERFGUN as described in [6] to create the index.

Given the value of `plain` hypotheses created by *table normalization*, we check the three indexes related to the language the hypothesis is based on for all resources this value could refer to and order the results by their frequency. For the top 10 entities, properties, and classes we create cell-based hypotheses.

For example, given the *table normalization* hypothesis shown in Figure 7 created for the string `Obama` found in the example table (Table 1), we create the hypothesis shown in Figure 8. The hypotheses contain the type, the URI,

and the confidence value (which is the frequency value normalized by the sum of frequency values of all candidates) of the resource.

Fig. 7. H5 – A table normalization hypothesis related to the cell 1-1 (column-row) in the example table (Table 1).

```
"H5": {
   "created_by_task": "table_normalization",
   "hypothesis_name": "plain",
   "based_on_hypotheses": ["H3"]
   "value": "Obama",
   "refers_to_column": 1,
   "refers_to_row": 1,
}
```

Fig. 8. H6 – An entity linking hypothesis related to the example table (Table 1).

```
"H6": {
   "created_by_task": "entity_linking",
   "hypothesis_type": "resource",
   "based_on_hypotheses": ["H5"],
   "entity": "dbr:Barack_Obama",
   "confidence": 0.67
}
```

2.9 Literal Linking Task

This tasks links strings found in cells to entities identified in other cells of the same row, thereby also identifying the property that links the entity with the literal value. That means, instead of linking strings to entities as done by *entity linking*, strings are linked to literal values in DBpedia. Therefore, it processes hypotheses created by *table normalization* (all types of hypotheses created by that task) and *entity linking*. For example, given Table 1, given the *table normalization* hypothesis shown in Figure 6 (which expresses that the string `4 August 1961` represents the literal `"1961-08-04"^^xsd:date`), and given the *entity linking* hypothesis shown in Figure 8 (which expresses that the string `Barack Obama` can be linked to the entity `dbr:Barack_Obama`), the hypothesis shown in Figure 9 is created. It expresses that the date is the birth date of Barack Obama.

This task makes use of an index for each language (containing language-tagged strings as well as datatyped-literals from the respective language version of DBpedia) to quickly retrieve a set of properties given an entity, a literal, and a datatype. When building this index, all properties that are used when building the *entity linking* indexes are ignored to avoid the creation of hypotheses similar to those created by *entity linking*.

Fig. 9. H7 – A literal linking hypothesis related to the example table (Table 1).

```
"H7": {
   "created_by_task": "literal_linking",
   "based_on_hypotheses": ["H3","H6"],
   "modified_literal": "1961-08-04",
   "property": "dbo:birthDate"
}
```

3 Statistics from Analyzing 1.000.000 Web Tables

From the 99 tar archives of the WTC dataset we select the first $62,500$ tables from each of the first 16 archives, thus resulting in a set of $1,000,000$ tables.

In a complete run over the corpus with the *language detection* task only, we found one of the following five languages (English, German, Catalan, French, Spanish) in most tables. Currently we exclude tables in other languages after the language detection step. These languages appear in the language statistics but not in statistics of later tasks such as *table normalization* and *entity linking*.

Note that in this section we do not evaluate the correctness of the hypotheses created. Rather, we provide data that might tell us something about the nature of the data and our approaches. For example, it is interesting to know how many tables exist where no hypotheses were added or where for a cell multiple literal linking hypotheses were created that relate the literal to multiple entities detected in other cells. For example, a value could be the birth date of one entity as well as the foundation date of another entity. Interesting tables can then be analyzed manually to check whether the tasks need to be improved or to devise more advanced tasks, such as triplification.

Scheduling Task: We measured computation time and the number of hypotheses created by each task. The complete processing of a table took an average of $0.9s$ ($\pm 116.1s$). Average values for other tasks are: $0.00005s$ ($\pm 0.001s$) for *table exclusion*, $0.014s$ ($\pm 0.015s$) for *language detection*, $0.003s$ ($\pm 0.2s$) for *table normalization*, $0.005s$ ($\pm 0.36s$) for *entity linking*, and $3.2s$ ($\pm 213.5s$) for *literal linking*. Tasks that transform WTC data into hypotheses such as *table classification*, *orientation detection*, and *table segmentation* are performed by the scheduler and were not measured individually.

The processing of $1,000,000$ tables took $266h$. *Table exclusion* took $1min$, *language detection* took $1.3h$, *table normalization* took $14min$, *entity linking* took $31min$ and *literal linking* took $264h$. Per table, *language detection* created an average of 1.1 hypotheses (± 0.4), *table normalization* created an average of 220 hypotheses (± 1360), *entity linking* created an average of 270.6 hypotheses (± 2117), and *literal linking* created an average of 0.01 hypotheses (± 0.4). For the tasks *orientation detection*, *table classification*, and *table segmentation*, for each table one hypothesis was created. Relative to the number of cells in a table, *table normalization* created 2.3 hypotheses (± 1.02), *entity linking* created 2.7 hypotheses (± 2.77), and *literal linking* created 0.00007 hypotheses (± 0.002).

Table Segmentation: In our sample of $1,000,000$ tables, we detected headers for $332,676$ tables. Note that header detection happened after the exclusion based on table type. Therefore, the $332,676$ tables with a header account for 95% of the tables that were not excluded.

Table Exclusion: 650,716 out of 1,000,000 tables were excluded because of the table type. There were no occurrences of exclusion after *language detection* or *entity linking*. The remaining 349,284 tables went through all processing steps.

Language Detection: From all the tables that were not excluded, English was detected for 321,066 tables (91.9%). The next most frequently occurring languages were German (20,464 / 5.8%), Catalan (7,248 / 2.1%), French (5,568

/ 1.6%), and Spanish (4697 / 1.3%). While for most tables we detected only one language, for 11.2% of tables we detected multiple languages. The most frequent combination is English and German, accounting for 32.4% of tables with multiple languages. Other frequent combinations are English and Catalan (7.5%) and English and French (3.1%).

Table Normalization: The task considered 316,006 Web tables and generated at least one hypothesis on 316,006 tables (100%), due to the plain hypothesis. We created a total of 35,433,908 hypotheses (including the `plain` hypotheses) with an average of 1.12 hypotheses created per cell. 572 hypotheses are related to *kg*, 1,587 to *km*, 6,378,332 are hypotheses on integers and 693,481 are hypotheses on floats. 839,459 is the total number of hypotheses generated for dates, which is split into 367,836 for the `Mon DD, YYYY` form, 62,656 for the `YYYY-MM-D` form, 139,659 for the `DD.MM.YYYY` form, and 269,308 for the `MM.DD.YYYY` form.

Entity Linking: We measured how many entity linking hypotheses (distinguished between entities, classes, and properties) were created on average per table, row, column, and cell. The results are shown in Table 3.

Table 2. Average number of entity linking hypotheses per table/row/column/cell distinguished by type (entity, class, property).

	entities	classes	properties
table	271.74	0.41	1.04
row	18.66	0.03	0.07
column	50.70	0.08	0.19
cell	3.21	0.005	0.01

Literal Linking: The task processes only tables with at least one *entity linking* hypothesis. For these tables, on average 0.08 literals per row were linked and in average 0% of the literals are related to at least two different entities. At least one hypothesis was generated for 556 tables (0.2%). We found that the top 5 properties that link literals to entities are `dbp:dateOfBirth`, `dbp:birthDate`, `dbo:percentageOfAreaWater`, `dbp:released`, and `dbo:birthDate`. For all these properties the object is either a numerical value or a date. Our current implementation might be too restrictive to match strings.

4 Related Work

In this section we discuss works related to the tasks that we described. Often, related approaches address more than one task and also address higher-level tasks, such as [10, 11] which present an approach for joint inference using Probabilistic Graphical Models and detect relations between columns. [15] extracts information from Wikipedia tables to populate a knowledge base. Their approach is based on extracting binary relations between entities.

Table Orientation Detection: Orientation detection in [14] is based on the intuition that if rows are similar to each other, then the orientation is vertical and if columns are similar to each other, then orientation is horizontal. The

authors introduce a distance metric for cells and use it to define distance metrics for rows and columns.

Table Classification: In [19] the authors distinguish between *genuine* tables where *a two dimensional grid is semantically significant in conveying the logical relations among the cells* and *non-genuine* tables. They define a set of features (layout features, content type features, and word group features) and experiment with decision tree classification and SVM. [3] propose a fine-grained taxonomy of HTML tables that contain relational knowledge. The table types were manually created after inspecting a set of tables and the authors analyzed the distribution of these types by manually classifying tables via statistics features, cell features, layout features, predicate features, and score features with which they train a classifier. The authors of [9] classify tables into the classes *Genuine Table with Header*, *Genuine Table without Header*, and *Non-genuine Table*.

Table Segmentation: [17, 5, 13] introduce Minimum Indexing Point Search and identify row and column headers by locating the minimum indexing point of a table. [2] analyze Web spreadsheets and perform a CRF-based segmentation.

Table Normalization: To detect the content of a cell, in [7] regular expressions are used. In [16] data types (i.e., string, numeric values, time-stamps, and coordinates) are detected via regular expressions. Handcrafted transformation rules are used to transform abbreviations, e.g., `Co.` to `Company`.

Entity Linking: [12] uses DBpedia as a knowledge base to map cell values. They built classifiers that pick the most likely entity from top N candidates. Similar to their work, we use DBpedia as a knowledge base to find the candidate entities for a cell value. Authors in [1] present an approach for linking the cell values to YAGO entities using an Iterative Classification Algorithm [8].

Literal Linking: [15] extract relations between pairs of cells and classify them into the relation between the two cells and the connection between the cell types and the possible relations for those.

5 Conclusions and Future Work

In this paper we present basic tasks that enrich a large subset of tables of an existing dataset with hypotheses based on DBpedia, we present statistics about the data and the hypotheses and make the corpus available to the community. We believe that this data allows others to focus on higher-level tasks of table understanding such as column understanding and to investigate how much information extracted from Web tables could be added to DBpedia.

Given that the WTC corpus contains 233 million tables and so far we annotated only one million tables, for future work we plan to annotate the entire corpus and support more than 5 languages by creating more regular expressions to detect and normalize values (e.g., for currencies and time measurements), and to develop tasks that build row-based and column-based hypotheses based on the cell-related hypotheses, such as for table orientation detection and table segmentation. More files will be available from our website in the future.

Acknowledgements

This work was supported by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG). The work was partially created within the Intelligent Systems Master students project *Information Extraction from Web Tables* at Bielefeld University under the supervision of B. Ell and S. Hakimov.

References

1. C. S. Bhagavatula, T. Noraset, and D. Downey. TabEL: Entity Linking in Web Tables. ISWC '15, pages 425–441. Springer, 2015.
2. Z. Chen and M. Cafarella. Automatic Web Spreadsheet Data Extraction. SSW '13, pages 1:1–1:8. ACM, 2013.
3. E. Crestan and P. Pantel. Web-scale Table Census and Classification. WSDM '11, pages 545–554. ACM, 2011.
4. J. Eberius, K. Braunschweig, M. Hentsch, M. Thiele, A. Ahmadov, and W. Lehner. Building the Dresden Web Table Corpus: A Classification Approach. BDC '15.
5. D. W. Embley, S. Seth, and G. Nagy. Transforming Web Tables to a Relational Database. ICPR '14, pages 2781–2786, Aug 2014.
6. S. Hakimov, H. t. Horst, S. Jebbara, M. Hartung, and P. Cimiano. Combining Textual and Graph-Based Features for Named Entity Disambiguation Using Undirected Probabilistic Graphical Models. EKAW '16, pages 288–302. Springer, 2016.
7. W. Holzinger, B. Krüpl, and M. Herzog. Using Ontologies for Extracting Product Features from Web Pages. ISWC '06, pages 286–299. Springer, 2006.
8. Q. Lu and L. Getoor. Link-based Classification. ICML '03, pages 496–503, 2003.
9. W. Lu, Z. Zhang, R. Lou, H. Dai, S. Yang, and B. Wei. Mining RDF from Tables in Chinese Encyclopedias. NLPCC '15, pages 285–298, 2015.
10. V. Mulwad, T. Finin, and A. Joshi. Automatically Generating Government Linked Data from Tables. In *AAAI Fall Symposium*, volume 4, 2011.
11. V. Mulwad, T. Finin, and A. Joshi. Semantic Message Passing for Generating Linked Data from Tables. ISWC '13, pages 363–378. Springer, 2013.
12. V. Mulwad, T. Finin, Z. Syed, and A. Joshi. Using linked data to interpret tables. COLD '10, pages 109–120. CEUR-WS.org, 2010.
13. G. Nagy, S. Seth, and D. W. Embley. End-to-End Conversion of HTML Tables for Populating a Relational Database. IAPR '14, pages 222–226, April 2014.
14. A. Pivk, Y. Sure, P. Cimiano, M. Gams, V. Rajkovic, and R. Studer. Transforming Arbitrary Tables into F-Logic Frames with TARTAR. *DKE*, 60(3):567–595, 2007.
15. C. Ran, W. Shen, J. Wang, and X. Zhu. Domain-Specific Knowledge Base Enrichment Using Wikipedia Tables. ICDM '15, pages 349–358. IEEE, 2015.
16. D. Ritze, O. Lehmberg, and C. Bizer. Matching HTML Tables to DBpedia. WIMS '15, 2015.
17. S. Seth and G. Nagy. Segmenting Tables via Indexing of Value Cells by Table Headers. ICDAR '13, pages 887–891, Aug 2013.
18. J. Wang, H. Wang, Z. Wang, and K. Q. Zhu. Understanding Tables on the Web. ER '12, pages 141–155. Springer, 2012.
19. Y. Wang and J. Hu. Automatic Table Detection in HTML Documents. *Series in Machine Perception and Artificial Intelligence*, 55:135–154, 2003.