Web Table Column Categorisation and Profiling

Oliver Lehmberg, Christian Bizer
Data and Web Science Group, University of Mannheim, Germany
{oli, chris}@informatik.uni-mannheim.de

ABSTRACT
Relational tables collected from HTML pages (“web tables”) are used for a variety of tasks including table extension, knowledge base completion, and data transformation. Most of the existing algorithms for these tasks assume that the data in the tables has the form of binary relations, i.e., relates a single entity to a value or to another entity. Our exploration of a large public corpus of web tables, however, shows that web tables contain a large fraction of non-binary relations which will likely be misinterpreted by the state-of-the-art algorithms. In this paper, we propose a categorisation scheme for web table columns which distinguishes the different types of relations that appear in tables on the Web and may help to design algorithms which better deal with these different types. Designing an automated classifier that can distinguish between different types of relations is non-trivial, because web tables are relatively small, contain a high level of noise, and often miss partial key values. In order to be able to perform this distinction, we propose a set of features which goes beyond probabilistic functional dependencies by using the union of multiple tables from the same web site and from different web sites to overcome the problem that single web tables are too small for the reliable calculation of functional dependencies.

1. INTRODUCTION
The integration of data collected from HTML tables (“web tables”), is a quite active research area. Many approaches have been proposed to solve the tasks of searching for tables, matching their schema and entities, and combining their data into a single output table [1, 4, 6, 17]. Further, web tables are being used to enrich existing knowledge bases or to construct new ones [5].

As recently shown in [13], web tables contain facts about a variety of entities and attributes that exist in knowledge bases such as DBpedia [11]. But, an even larger amount of data from web tables cannot be matched to the existing schemas of knowledge bases and is thus a candidate for schema extension, meaning that additional, previously unknown attributes are added to an existing schema. Current methods operate under the assumption that web table data has the form of binary relations, which describe the relationship between two entities or an entity and a value. Our exploration of the content of web tables, however, shows that web tables contain quite a number of different types of relations, some of which are not suitable for the application of the existing methods.

Most of the approaches that assume binary relations, apply an entity column detection heuristic [15], such as “use the most unique string column”, or machine-learning methods to find a single entity column. The entity column is an approximate key which names the entities that are described by the rows in the table and determines the topic of the table. Figure 1 (a) shows an example. In this table, the column labeled “Athlete” is the entity column. Based on the entity column, the “Nat” column can be used to extract a binary relation from the table, namely the nationality of the given athletes. But, the column labeled “Mark” depends not
only on the entity column, but also on the event in which this mark was achieved. So, extracting a binary relation in the form of Mark(Philips Dwight, 8.31) would be an error.

To distinguish between binary and non-binary relations, functional dependencies can be determined from the data. If a functional dependency holds between the entity column and another column, this means that the column’s value is always determined by the entity column and that there exists a binary relation. If no such dependency holds, it must be a non-binary relation, meaning additional factors, such as another entity column, determine the value or it is not related at all.

However, web tables introduce additional challenges which render traditional functional dependencies indiscriminative. First, the entity column is only an approximate key and is not necessarily unique, which is why probabilistic functional dependencies should be preferred. Second, partial key values are required to detect a functional dependency with more than one column as determinant. But parts of the key values are often located on the web page surrounding the table and not in the table itself, for example the title of the table in Figure 1 (a). And third, web tables are small and in most cases there is not enough data to make a decision about the actual dependencies. For example, the table in Figure 1 (a) contains each athlete only once, but the web site contains many similar tables with sports results. Only if we look at multiple tables at once, we can infer that “Nat” is a binary relation while “Mark” is not.

The contributions of this paper are that 1) we propose a novel categorisation scheme for web table columns, 2) we present features going beyond probabilistic functional dependencies that leverage the large number of web tables to distinguish between the top-level categories of the proposed categorisation schema and 3) we show, based on the analysis of a large corpus of web tables, that a large fraction of the columns in web tables are non-binary relations, which violates the assumptions of many existing algorithms.

2. COLUMN CATEGORISATION

Figure 2 shows an overview of the categories that we propose. The categories are intended to describe the columns and their relation to the detected entity column, i.e., a column can have a different category if a different entity column is chosen. This simply accommodates the fact that the entity column determines how the data is used in existing approaches.1

The primary distinction is between columns which represent a binary relation or an n-ary relation. A binary relation holds between two, an n-ary between more than two arguments. A column can also be independent, which corresponds to any relation that does not include the entity column (but can otherwise include any number of arguments). In general, binary relations are suitable for methods that add data to knowledge bases, which usually model their facts in the form of triples (subject-predicate-object statements). N-ary relations cannot be directly represented and are hence less suitable.

Binary Relation. A relation that holds between the entity column and the values in the column. A binary relation

---

1It can, for example, determine the class in a knowledge base which is considered for extension with the data from the table.

2Nevertheless, there are methods which use the time information as metadata of their relations, such as the work of Zhang et al. [18]. But similar to additional entity columns, the time information is often not present in the table itself.
rank columns do not contain additional data about the entities. An additional entity column is part of an n-ary relation. It determines the value of the relation in combination with the entity column (and maybe more additional entities). An example can be seen in Figure 1 (c), where “From” and “By Airlines” are additional entity columns which together with the entity column “To” determine the value of “USD $ Fare”. 

Others describes all n-ary relations which are neither a rank nor an additional entity column, i.e., comprises all values that depend on multiple entities.

**Independent Column.** A column that has no direct relation to the detected entity column. Although such a column can contain meaningful data, it is not related to the detected entity column. The first subcategory is empty, which describes columns that do not contain any data. Horizontally stacked columns emerge when a website designer aligns two tables next to each other using the same HTML table tag. The result is that one row contains multiple tuples, which have no relation other than being placed together for visual reasons. An example is shown in Figure 1 (b). The same idea applies to list columns which appear in tables that only contain the entity column multiple times. The sole purpose of such tables is to list a set of entities, hence they do not contain any relations (see Figure 1 (d)). Another subcategory is layout, which describes columns that are inserted into tables for formatting reasons or contain navigational content (see the last column in Figure 1 (c), which only provides a link to another page on the same website but does not contain actual data). The last subcategory is indirect columns, which are transitively dependent on the entity column, but not directly.

### 3. COLUMN MATCHING

In this Section, we introduce the data source for our experiments and describe how correspondences between different web tables are created. These correspondences enable us to create the union of multiple web tables. We use the Web Data Commons Web Tables Corpus 2012.\(^3\) We remove non-relational, non-English and very small tables from the corpus and apply the T2K Matcher [12] to obtain the entity columns as well as class, attribute and entity correspondences to the DBpedia knowledge base in order to understand the content of the tables. After removing exact duplicates, 503,104 unique tables remain. For details concerning the corpus and the matching results we refer to Ritze et al. [13].

If we consider such a large corpus of web tables, we will obviously find multiple columns that have the same meaning, or represent the same attribute. For most tasks, we are interested in obtaining frequent attributes that appear in many tables. This is motivated by the goals of having a high coverage of entities, having more evidence for choosing the correct values in a subsequent data fusion step, and having a general indication for the relevance of the attributes. In this context, an attribute is represented by one or more columns in our web tables dataset.

Attributes that exist in the knowledge base are represented by all columns with correspondences to these attributes, as determined by the T2K Matcher. An example for such an attribute, which is described by its class and attribute names from DBpedia, would be “dbp:Country/dbp:populationTotal”. For columns where no matching attribute was found in the knowledge base, we create correspondences between all columns with the same column header and data type, whose tables have correspondences to the same class. For example, the attribute “dbp:Country/sales tax rate/numeric” is represented by all columns from tables with a correspondence to the country class that contain numeric values and have the column header “sales tax rate”. With this basic but strict approach, we want to avoid creating correspondences between unrelated columns and accept that we will potentially create multiple attributes with the same meaning.

Table 1 shows the number of columns in our dataset and how many attributes were created. One entity column was detected for each table. Besides, about 400k columns have correspondences to DBpedia, resulting in roughly three thousand attributes. The attributes represent 721 unique properties from the DBpedia ontology namespace combined with the respective class that was assigned to the tables (so dbp: populationTotal for countries and for cities are two different attributes). More than 2 million columns, which represent 300k attributes, have no correspondences in the knowledge base.

### 4. CATEGORY GOLD STANDARD

We randomly sample 400 web tables (about 1,400 columns) from the corpus described in Section 3 and manually annotate them with the categories described in Section 2. We exclude 13 tables because they were matched incorrectly or we could not understand the content due to foreign languages.\(^4\)

For the web tables in our sample, we add the following conditions to ensure that the data is useful for a task like schema extension, where evidence from multiple sources is required for reliable results. Each web table must have a schema which appears on at least two different web sites. The schema of a web table in this context is the ordered set of column headers in combination with the column data types. Further, the union of the entities of all tables having this schema must contain at least 50 entities and at least three of these entities must appear in tables from multiple web sites.\(^5\) The union of entities is determined using the entity correspondences generated by the T2K Matcher. Note that we only consider a single column as entity column for each table. In cases where multiple columns form the primary key, we categorise all other columns as n-ary relations.

Table 2 shows the distribution of the column categories in our sample. Binary relations account for 42% of the sample, 46% are n-ary and 12% are independent.

\(^3\)http://webdatacommons.org/webtables/

\(^4\)There are obviously cases where our “English tables” heuristic [12] made a mistake.

\(^5\)These thresholds have been chosen empirically to ensure a certain level of overlap in the data while not excluding too many of the tables in the corpus.

<table>
<thead>
<tr>
<th>Category</th>
<th>Number of Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns total</td>
<td>4,259,258</td>
</tr>
<tr>
<td>Columns with correspondences</td>
<td>918,769</td>
</tr>
<tr>
<td>Entity columns</td>
<td>503,104</td>
</tr>
<tr>
<td>Non entity columns</td>
<td>415,665</td>
</tr>
<tr>
<td>Attributes</td>
<td>3,096</td>
</tr>
<tr>
<td>Columns with no correspondences</td>
<td>2,340,489</td>
</tr>
<tr>
<td>With headers</td>
<td>2,069,815</td>
</tr>
<tr>
<td>Unique headers</td>
<td>174,151</td>
</tr>
<tr>
<td>Attributes</td>
<td>300,272</td>
</tr>
</tbody>
</table>
Another column A, which capture the probability of the dependency, should functional dependencies (pFDs) as proposed by Wang et al. to calculate functional dependencies. Hence, probabilistic ambiguous and must be interpreted before they can be used intranet environment. For example, entity labels can be to expect a higher level of uncertainty as compared to an → X all (independent columns).

Binary 31.26% Time-independent 31.26% Time-varying 10.43%

N-Ary 45.81% Others 34.39% Rank 8.74% Additional entity column 2.68%

Independent 12.47% Horizontally stacked 5.00% Empty 4.23% List 1.90% Indirect 0.78% Layout 0.56%

Table 2: Distribution of Column Categories in the Gold Standard

One interesting observation we can make from this result is that additional entity columns appear very infrequently in the web tables, compared to the high number of columns which are n-ary relations. This means, even if we were trying to detect composite entity columns instead of single entity columns, only in very few cases we would succeed. We can assume that the additional entities must be somewhere outside the table but on the web page containing the table (otherwise not even a human could interpret the values). A second observation is that transitive dependencies (category “indirect”) are also quite rare, and in the cases that we encountered, they could have been avoided completely by choosing a different entity column.

5. CLASSIFICATION

Knowing the column categories and their distribution in our sample, the question arises whether they can be automatically detected. This allows us to categorise our full dataset and profile the web tables corpus with respect to the types of relations. We restrict ourselves to the top-level categories, which are crucial to determine whether a column is generally suitable for the existing methods or not.

Our approach is to analyse the values and calculate dependencies from the data. All values in columns which are binary relations must be determined by the values of the entity column. If not, they either depend on additional entities (n-ary relations) or do not depend on the entity column at all (independent columns).

5.1 Features

The fact that column X, the entity column, determines another column A is expressed as functional dependency X → A. However, in the context of web tables, we have to expect a higher level of uncertainty as compared to an intranet environment. For example, entity labels can be ambiguous and must be interpreted before they can be used to calculate functional dependencies. Hence, probabilistic functional dependencies (pFDs) as proposed by Wang et al. [16], which capture the probability of the dependency, should be preferred.

The probability that a pFD holds with respect to a specific entity V_X of entity column X depends on how often the most frequent value V_A of column A for this entity occurs, expressed as |V_A, V_X|, and the number of tuples about entity V_X, expressed as |V_X| in Equation 1.

\[
Pr(X \rightarrow A, V_X) = \frac{|V_A, V_X|}{|V_X|}
\] (1)

The probability that the pFD holds for the whole table R is then calculated by taking the average over all distinct entity column values D_X as shown in Equation 2.

\[
Pr(X \rightarrow A) = \frac{\sum_{V_X \in D_X} |V_A, V_X|}{|D_X|}
\] (2)

A pFD close to 1 is an indication for a binary relation, while a pFD close to 0 indicates an n-ary relation or independence. In practice, however, most web tables are too small for this assumption to hold. As a pFD can only decrease if an entity occurs multiple times and with different values, tables that contain each entity only once will not be decidable.

As a remedy, we make use of the large number of tables in our corpus. By combining the data of multiple tables, we increase the chance of observing an entity multiple times, which increases the chance to calculate meaningful pFDs. We create the union of all tuples with the attribute that is represented by column A (attributes are generated as described in Section 3) and calculate the pFD based on the result.

But, due to the high level of heterogeneity in web data, we might still run into problems. It would be inappropriate to expect that the creators of different web sites all agree on the same value formatting and have only correct data from the same original source. For example, different web sites listing the population of countries likely have data from different points in time and with a different precision. Concerning the data from a single web site, however, these assumptions are much more likely to be true. As an example, consider the “Nat” column from Figure 1 (a). This column is repeated in every table that contains sport results and always has the same value formatting.

For this reason, we create different pFD measures: intra- and inter-PLD pFD. Intra-PLD pFDs are calculated on all the tuples with the attribute that is represented by column A from an individual web site. Inter-PLD pFDs are calculated on the corresponding tuples that appear on different web sites. While the inter-PLD pFD measure is likely the noisiest, it is also the one with the best chance to capture the actual dependencies which are approximated by pFDs. By adding a similarity function and a similarity threshold, we can allow for more variation in the values, resulting in fuzzy probabilistic functional dependencies (fpFDs). The calculation is the same as for pFDs, with the difference that the most frequent value V_X is determined by applying a clustering algorithm to all values first and then using the size of the largest cluster as |V_A, V_X|. To be specific, we calculate the connected components over the graph of all values as nodes, where edges are added between two values if their similarity given by the similarity function is above the similarity threshold.

Table 3 summarizes the features that we create from these various FDs. Features (1) and (3) are the intra- and inter-PLD pFDs and feature (8) is the fuzzy pFD as described.

---

\[Equations 1 and 2 directly adopted from Wang et al. [16].\]

\[A pay-level domain (PLD) refers to the part of a domain name that is paid for and is intended to capture the notion of a web site, i.e., all pages under the same PLD are assumed to belong to the same web site.\]

---
above. Feature (2) is the average over all intra-PLD pFDs of an attribute, which describes the average probability of the dependency on all involved web sites. Features (4) and (5) set the inter-PLD pFD of the column in relation to the detected entity column and features (6) and (7) set it in relation to all other columns in the table.

We add additional features as indicators for specific sub-categories. The detection of empty columns in feature (9) mainly involves removing encoding artifacts from the HTML-based representation (“&nbsp; etc.). Tables that only list entities are often sorted alphabetically either from left to right or top to bottom, which is captured by feature (10). Feature (11) is an indicator for horizontally stacked tables, i.e., multiple tables in the same HTML table tag, where the column headers are repeated in the exact same order and usually only empty columns are in between. Finally, feature (12) is an indicator for ranks, which often have the same column headers.

### Table 3: Features used for Column Classification

<table>
<thead>
<tr>
<th>#</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intra-PLD pFD</td>
<td>The pFD calculated on all values from the same PLD</td>
</tr>
<tr>
<td>2</td>
<td>Avg. Intra-PLD pFD</td>
<td>The average of (1) over all PLDs of an attribute, weighted by entity count</td>
</tr>
<tr>
<td>3</td>
<td>Inter-PLD pFD</td>
<td>pFD calculated on values from multiple web sites</td>
</tr>
<tr>
<td>4</td>
<td>Entity column Inter-PLD pFD</td>
<td>The (3) of the entity column of the table</td>
</tr>
<tr>
<td>5</td>
<td>Entity column diff.</td>
<td>The difference between (3) and (4)</td>
</tr>
<tr>
<td>6</td>
<td>Avg. Table Inter-PLD pFD</td>
<td>The average of (3) of all attributes in the table</td>
</tr>
<tr>
<td>7</td>
<td>Table Inter-PLD Ratio</td>
<td>The ratio between (3) and (6)</td>
</tr>
<tr>
<td>8</td>
<td>Fuzzy pFD</td>
<td>The pFD of all values</td>
</tr>
<tr>
<td>9</td>
<td>Is Empty</td>
<td>1 if the column contains no characters except spaces</td>
</tr>
<tr>
<td>10</td>
<td>Is List</td>
<td>1 if all values in the table are sorted horizontally or vertically</td>
</tr>
<tr>
<td>11</td>
<td>Is Horizontally Stacked</td>
<td>1 if all column headers in the table are repeated in the exact same order</td>
</tr>
<tr>
<td>12</td>
<td>Is Typical Rank</td>
<td>1 if the column header is “rank”, “#” or “pos”</td>
</tr>
</tbody>
</table>

### 5.2 Evaluation

We apply the one-versus-rest strategy and learn linear regressions for each top-level category and use the manually annotated tables to evaluate the performance of the features described above. For training, we use 192 tables with 699 columns and for testing the remaining 195 tables with 703 columns. To classify a column, all regressions are applied and the prediction with the highest confidence is used.

Table 4 shows the resulting confusion matrix. Overall, we achieve an accuracy of 74.54% and precision and recall values between 67% and 86% for all classes. The features with the strongest impact for the binary and n-ary categories are (3)-(7) and for the independent category (9)-(11). As a baseline, we use only feature (1) and achieve precision and recall values of 49%/68% for binary and 58%/41% for n-ary.

#### Table 4: Confusion Matrix for the Column Category Classifier

<table>
<thead>
<tr>
<th>Category</th>
<th>Attribute Freq. %</th>
<th>Column Freq. %</th>
<th>Class recall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>true binary</td>
<td>true n-ary</td>
<td>true ind.</td>
</tr>
<tr>
<td>pred. binary</td>
<td>219  92</td>
<td>16</td>
<td>71.33%</td>
</tr>
<tr>
<td>pred. n-ary</td>
<td>74   252</td>
<td>9</td>
<td>75.22%</td>
</tr>
<tr>
<td>pred. ind.</td>
<td>2    6</td>
<td>53</td>
<td>86.80%</td>
</tr>
<tr>
<td>total</td>
<td>295</td>
<td>353</td>
<td>61.95%</td>
</tr>
</tbody>
</table>

#### Table 5: Distribution of Column Categories in the Corpus

There is an equal amount of columns which are incorrectly classified between binary and n-ary. Reasons can be the following. Either there is not enough overlap in the entities and their values, such that the calculated dependencies are high, leading to a prediction of binary for an actually n-ary relation. Or the values are too different for a binary relation, such that the calculated dependencies are low, leading to a prediction of n-ary. An example would be the attribute “nationality” where multiple surface forms of the value “Germany” could be “GER”, “DE” or an image depicting the flag of Germany. Of course, matching errors also lead to incorrectly estimated dependency values and can cause the same problems.

### 6. CORPUS PROFILING

We run our classifier on all columns that belong to an attribute which has an overlap in its entities on at least two different web sites (PLDs). This dataset contains a total of 2,259,238 columns, which by their correspondences can be merged to 66,999 attributes.

Table 5 shows the distribution of categories in terms of columns and attributes (columns merged by their correspondences). Binary relations account for about one fifth of all attributes (21.7%) while the majority is classified as n-ary (61.9%) and independent ones amount to 16.4%. Table 6 shows examples of attributes for several DBpedia classes that were classified as binary relations.

The difference between the number of attributes and the number of predictions for attributes is to be expected. It results from the fact that not all columns that belong to an attribute are classified consistently. However, the rather low difference indicates that our strict column matching approach did not group too many unrelated columns.

We assume that the high percentage of n-ary relations is created by the large amount of tables with sports results, which according to the class distribution are very frequent in the corpus. Earlier work of Ritze et al. [13] has shown that a large proportion of web tables is matched to athletes, which are usually mentioned in tables with the results of sport events, which are obviously n-ary relations.

#### Estimation of result quality

The fundamental requirement of our approach is the existence of overlap between the web tables in our corpus. But, as we know from earlier results [13], there is a large fraction of infrequent entities in the tables. To get an idea how strongly this affects
our results, we can estimate the recall of binary relations on the full dataset. A column with a correspondence to the knowledge base is supposed to be a binary relation, as the attributes in the knowledge base are binary relations.\(^9\) As an estimation of the recall, we determine the fraction of columns that were classified as binary relation of all columns with schema correspondences. To see the impact of the overlap, we estimate this recall on two datasets: One where we apply the constraints mentioned in Section 3 and the one which we described earlier in this Section. On the constrained dataset, the estimated recall is about 67%. On the full dataset, the estimation only reaches about 41%. This result indicates that there is not enough overlap in the corpus for many columns or that the values are more heterogeneous than expected. However, on the subset, which is preferred for more reliable results anyway, the achievable performance is much higher.

<table>
<thead>
<tr>
<th>Class</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BaseballPlayer</td>
<td>pos, infield, team, hit, wt, no</td>
</tr>
<tr>
<td>Company</td>
<td>symbol, market cap, revenues, profits, assets, sales, employees</td>
</tr>
<tr>
<td>Country</td>
<td>sales tax rate, voltage, freq</td>
</tr>
<tr>
<td>Device</td>
<td>resolution, colors, carrier, browsing time, music playback</td>
</tr>
<tr>
<td>EducationalInstitution</td>
<td>highest award, enrolled, state, top degree, website, % minorities, acceptance rate</td>
</tr>
<tr>
<td>Film</td>
<td>genre, year, (age) rating, studio, director, length</td>
</tr>
<tr>
<td>Fungus</td>
<td>shape, characteristic, hosts, context, lower surface, taste</td>
</tr>
<tr>
<td>HorseRace</td>
<td>age, distance, track, division, sex, surface, weight</td>
</tr>
<tr>
<td>Mineral</td>
<td>hardness, density, formula</td>
</tr>
<tr>
<td>Plant</td>
<td>color, life cycle, bloom season, bloom color, moisture, sun, vitamin a content, min temperature</td>
</tr>
<tr>
<td>Satellite</td>
<td>position, duration, mass, mission results, launch vehicle</td>
</tr>
<tr>
<td>Software</td>
<td>version, license, windows, platform, mac os, latest stable</td>
</tr>
</tbody>
</table>

Table 6: Frequent Attributes classified as Binary Relations

7. RELATED WORK

A related categorisation scheme has already been proposed by Crestan and Pantel [3]. They define different types of web tables and differentiate on the highest level between tables that contain relational knowledge and tables which are used for layout purposes. Our categorisation scheme can be understood as an extension which describes the columns of tables that contain relational knowledge.

The work of Wang et al. [16] provides the notion of probabilistic functional dependencies which is the foundation of our classification approach. Similar to our approach, they calculated the dependencies on data merged in a mediated schema, but they did not make the distinction between sources (i.e., different web sites) where an exact comparison of values is promising or where a similarity measure should be applied. The original application was to use the pFDs to find “dirty” data sources and for the normalisation of their large, mediated schemas. Their normalisation aimed at splitting tables with transitive dependencies, while we are interested in attributes which depend on multiple entities.

A scalable method for the efficient calculation of functional dependencies is provided by Heise et al. as well as Papenbrock et al. [9, 14], but they only consider exact functional dependencies. Also, due to the fact that most partial keys are located outside of the web tables, the calculation of all possible dependencies in a single table is not very promising in our case.

Concerning the extension of data sources, approaches such as the works of Cafarella et al. [1, 2], Das Sarma et al. [4] or Lee et al. [10] try to find matching web tables and rank them by their entity overlap or attribute co-occurrence probabilities. The work of Yakout et al. [17] performs the schema extension task on a given set of entities using correspondences between key/value collections generated from tables. However, these methods consider how related an attribute is to a class and do not distinguish between binary and non-binary relations. A comparable evaluation of such systems is tricky, as in most cases several annotators are asked to judge the usefulness or relatedness of the result. It is unclear whether “related” n-ary relations are treated as correct or incorrect result. We hope that our categorisation system can be used in future work to clarify this.

There are other approaches which use background knowledge to understand the content of web tables. Background knowledge is what a human applies as the table content and surrounding context are interpreted to understand the meaning of the columns. Venetis et al. [15] use Open Information Extraction methods to find potential relations on web pages and explicitly focus on binary relations. The work of Gupta et al. [6] describes a system that learns extraction patterns for attributes via distant supervision from Freebase and applies them on large text corpora. The discovered relations are then used to interpret the web tables. He et al. [8] use search engine query logs to determine the meaning of attributes in web tables. The work of Halevy et al. [7] shows how to understand complex attribute names and enables the detection of additional entities in the column header (their example is “tyre price in Singapore” for the class sports car). Such approaches can be combined with our method, as the understanding of the column headers can help with the matching of columns and hence increases the amount of data that can be used to calculate the features we proposed in this work.

8. CONCLUSION

The work we present here consists of three parts. First, we propose a categorisation scheme for web table columns which highlights the variety of data that any approach dealing with web tables has to cope with. We believe that by developing specific methods for the detection of individual categories, many of the existing approaches for the integration of web tables with other data can be improved. Second, we design a set of features for the detection of the top-level categories which are purely based on the data in the tables and go beyond the pure application of probabilistic functional dependencies by creating the union of tables from the same web site and from different web sites. Third, we apply our method to a large number of columns and show that a large number of n-ary relations exists, which violates the assumptions of many existing approaches that use web table data.

\(^9\)Which is true for most, but not all properties in the DBpedia ontology.
9. REFERENCES


