Stitching Web Tables for Improving Matching Quality

Oliver Lehmberg, Christian Bizer
Data and Web Science Group, Universität Mannheim
B6 26, 68159 Mannheim, Germany
{oli,chris}@informatik.uni-mannheim.de

ABSTRACT

HTML tables on web pages ("web tables") cover a wide variety of topics. Data from web tables can thus be useful for tasks such as knowledge base completion or ad hoc table extension. Before table data can be used for these tasks, the tables must be matched to the respective knowledge base or base table. The challenges of web table matching are the high heterogeneity and the small size of the tables.

Though it is known that the majority of web tables are very small, the gold standards that are used to compare web table matching systems mostly consist of larger tables. In this experimental paper, we evaluate T2K Match, a web table to knowledge base matching system, and COMA, a standard schema matching tool, using a sample of web tables that is more realistic than the gold standards that were previously used. We find that both systems fail to produce correct results for many of the very small tables in the sample. As a remedy, we propose to stitch (combine) the tables from each web site into larger ones and match these enlarged tables to the knowledge base or base table afterwards. For this stitching process, we evaluate different schema matching methods in combination with holistic correspondence refinement. Limiting the stitching procedure to web tables from the same web site decreases the heterogeneity and allows us to stitch tables with very high precision. Our experiments show that applying table stitching before running the actual matching method improves the matching results by 0.38 in F1-measure for T2K Match and by 0.14 for COMA. Also, stitching the tables allows us to reduce the amount of tables in our corpus from 5 million original web tables to as few as 100,000 stitched tables.

1. INTRODUCTION

Tables on web pages ("web tables") are an intriguing source of data for several reasons. They represent their content in a structured fashion, i.e., rows and columns are separated and may include column headers, but no explicit schema information is provided. Further, they exist in large quantities on the Web and cover many different topics [13]. This makes them an interesting data source for large-scale data extraction and integration, as required for the construction or completion of knowledge bases [11, 29] or the ad-hoc extension of arbitrary base tables with additional attributes [7, 20, 34].

Before table data can be used for these tasks, the tables must be matched to the respective knowledge base or base table. This matching task is addressed by various existing methods [21, 28, 31, 32, 35]. The main challenges of the task are the high heterogeneity of the web tables and their small size. For example, the 90 million relational tables in the Web Data Commons web tables corpus 2015 [19][1] have a median number of rows of 6. In contrast to this relatively small size of HTML tables on the Web, the gold standards that are used to evaluate web table matching systems, such as the T2D gold standard [28] or the Limaye gold standard [21], contain rather "ideal" web tables with respect to the size of the tables, with a median of 100 and 21 rows, respectively. The gold standards thus do not properly reflect the size distribution of the HTML tables on the Web.

Figure 1 shows examples of small web tables describing video games. We see how information about the games that likely resides in a single table in the database behind the website is published as a set of small tables on different pages.

In order to investigate whether small tables are correctly matched by current matching methods, we evaluate two existing matching tools, T2K Match [28] and COMA [1], using a sample of web tables that is more realistic than the tables in the T2D or Limaye gold standards. Our experiments show that both tools fail to produce an acceptable result on this sample.

To counter this problem, we propose to stitch web tables from the same website before running any of the existing matching methods. In this context, stitching means that we combine web tables based on a schema mapping. This is feasible, because many web tables are created systematically, for example by paging, where a long table is broken up into multiple parts, or by master-detail views. We build on the idea of Ling et al. [22], to combine small web tables into larger ones in preprocessing, and apply it to the task of web tables to knowledge base schema matching. We also extend the idea by an additional stitching step for web tables with different schemata. For T2K Match and COMA, we see improvements of 0.26 and 0.1, respectively, in F1-measure when creating the union of all tables from the same web

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

Proceedings of the VLDB Endowment, Vol. 10, No. 11

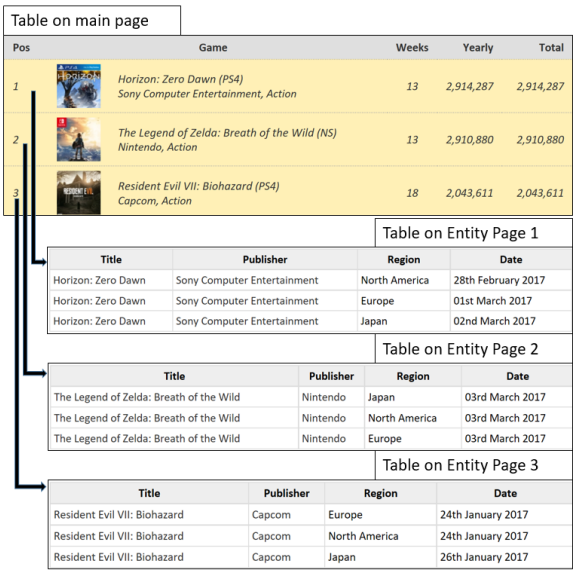


Figure 1: Example how information about video games is broken up into multiple small web tables on different pages.

site having the same schema. An additional improvement of 0.12 and 0.04, respectively, in F1-measure is achieved by further stitching union tables from the same web site having overlapping schemata. For individual web sites, we can see improvements of up to 0.76 in F1-measure.

The contributions of this paper are as follows:

1. We show that web table stitching strongly improves the results of existing schema matching methods on a more realistic sample of web tables than the currently used gold standards.
2. Our experiments show that creating the union of all web tables with the same schema from the same web site already results in a large improvement and should hence be done by all web table matching systems.
3. Through the experimental evaluation of standard and holistic matching approaches, we find that we can stitch union tables from the same web site with overlapping schemata at high precision, which results in an additional improvement of the matching results compared to only creating the union of tables with the same schema.

This paper is organised as follows: Section 2 introduces the use case of web table to knowledge base matching. Section 3 presents detailed schema statistics about the tables in our corpus. The effect of creating union tables is shown in Section 4. Sections 5 compares different matching methods for union tables. Section 6 evaluates the effect of using these methods for an additional stitching step on the overall matching performance. Section 7 investigates the impact of table stitching on the amount of correctly extracted data values and analyses how different properties of the web tables influence the achieved performance gain. We discuss related work in Section 8 and conclude with Section 9.

All methods were implemented using the *WInte.r* data integration framework. [2] The source code and all datasets that were used for the experiments in this paper are publicly available. [3]

[2]https://github.com/olehmberg/winter
[3]https://github.com/olehmberg/WebTableStitching

2. WEB TABLES TO KB MATCHING

This section introduces the use case of web table to knowledge base matching and establishes a performance baseline by matching a random sample of web tables from the Web Data Commons (WDC) web table corpus [19] against the DBpedia knowledge base [17] using T2K Match and COMA. In the following sections, we will repeat the same experiment in order to demonstrate the impact of table stitching on the matching performance.

Web table to knowledge base matching is a prerequisite for knowledge base augmentation (KBA), which is the task of filling missing values in a knowledge base or adding new properties or entities to the knowledge base [11,21,28,29,31, 32,35]. The goal of web table to knowledge base matching is to recognise entities and properties from the knowledge base in a given web table. The focus in this paper is matching the schemata of web tables to properties in a knowledge base.

Web Table Corpus. The experiments in this paper use a subset of the WDC web tables corpus 2015. This subset contains all 5 176 160 de-duplicated web tables from the corpus in which at least one entity from the DBpedia knowledge base could be recognised. Tables without any recognisable entity from the knowledge base cannot contribute to the knowledge base augmentation task and can hence be excluded. The detection of entities was performed by comparing the table values to the entity labels in DBpedia with Jaccard similarity and a threshold of 0.7. The tables in the subset have a median of 6 rows and originate from a total of 86 316 different hosts.

Random Sample. From this corpus of tables, we draw a random sample of 1 000 web tables. The sample contains web tables from 401 different hosts. In total, these hosts contribute 3.5 million web tables to the 5 million tables subset discussed above. The web tables in the 1 000 table sample have between 2 and 432 rows and a median of 4 rows. By manually matching the tables against the DBpedia knowledge base, we obtain a reference schema mapping consisting of 427 web table column to KB property correspondences. Out of these correspondences, 204 refer to the "rdfs:label" property (containing the names of the entities) and 223 to other DBpedia properties.

Gold Standards. Up till now, two gold standards were used to compare web table matching systems: The T2D gold standard[4] [28] contains 233 web tables that are mapped to the DBpedia knowledge base [17] and are annotated on class, property and entity level. The Limaye gold standard [21] contains 401 web tables that are mapped to the YAGO knowledge base. However, the web tables in these gold standards were selected using seed entities. This results in a selection of tables in which many entities from the knowledge base are found and which are generally larger to cover more entities. The tables in the T2D gold standard have a median length of 100 rows and the tables in the Limaye gold standard, as re-built by Bhagavatula et al. [4][5], have a median of 21 rows. However, the tables in the WDC web tables corpus 2015 [19], for example, have a median of only 6 rows.

T2K Match. We use T2K Match [28][6] as an example of a specialised web table to knowledge base matching sys-

[4]http://webdatacommons.org/webtables/goldstandard.html
[5]http://websail-fe.cs.northwestern.edu/TabEL/
[6]https://github.com/olehmberg/T2KMatch

tem in our experiments. T2K Match matches web tables to classes, table rows to entities and columns to properties in a target knowledge base. Before matching a web table, one column is determined to be the entity label column, i.e., the column that contains the names of the entities that are described in the table. The detection of the entity label column is performed heuristically, by choosing the most unique text column. Afterwards, candidate correspondences to entities and properties from the knowledge base are used to determine the majority class of the entities in the web table. Next, entity and property correspondences are iteratively refined, until a final mapping is created.

COMA. In order to test the general applicability of the stitching method, we repeat the experiment using COMA 3.0 [1, 10]. COMA is a general-purpose schema matching tool which exploits both the attribute labels and data values. As COMA is not tailored for the web table to knowledge base task, we change the experimental set-up in the following way: We focus on the schema matching part and match each web table to the schema of its corresponding class in the knowledge base. Due to limitations of the COMA application, we cannot use more than 1 000 data values per attribute and hence sample 1 000 entities from the corresponding DBpedia classes. We guarantee that the entities corresponding to those in the web tables are included, then add random entities until 1 000 entities are reached.

Experiment. We run T2K Match and COMA on the random sample of 1 000 web tables and find that the performance for the schema matching task is much worse than on the T2D gold standard. On the gold standard, T2K Match achieves an F1-measure of 0.7 for the schema matching task. On the sampled tables, however, the achieved F1-measure is only 0.24. The achieved precision is only 0.20 with a recall of 0.32 and 95 of the 204 mappable web tables are correctly identified. The reason for the low precision is that non-mappable tables are mapped to the knowledge base, i.e., a correspondence to "rdfs:label" is created. T2K Match incorrectly maps 444 tables to the knowledge base. Of these tables, 90% have only six rows or less. For the 109 web tables that were not mapped although it would have been correct, we also find indication that the size of the tables could be a problem: 52% of these tables have only up to six rows. The set of tables for which correct correspondences were found contains less than 35% tables having six rows or less. COMA achieves an F1-measure of 0.37 with a precision of 0.54 and a recall of 0.28. Other than T2K Match, COMA creates only 18 correct correspondences to the "rdfs:label" property, but 110 correct correspondences to other properties. It is noticeable that the correct correspondences are between columns and properties with similar or equal labels. This indicates that COMA did not consider the similarity of the data values to suffice in order to create correspondences in many cases.

This baseline experiment shows that both systems do not achieve satisfying results on matching the sampled tables.

3. WEB TABLE PROFILING

Most web sites use content management systems or other programs to generate HTML pages. The web tables on these pages are also automatically generated and the same schema is thus used by web tables on many different pages. In order to support this statement, we investigate the frequency

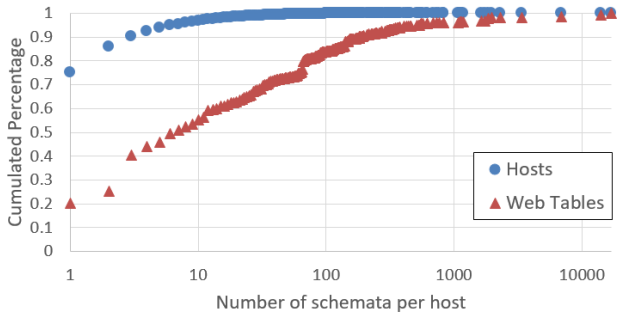


Figure 2: Number of schemata per host.

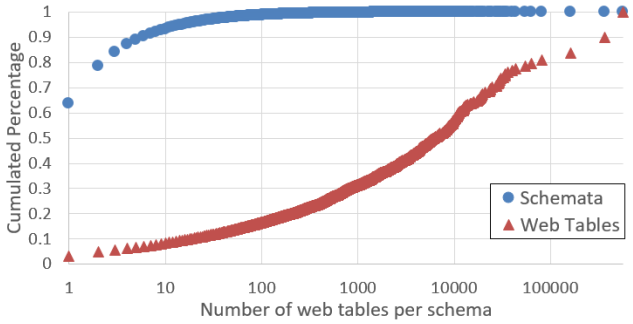


Figure 3: Number of web tables per schema.

distribution of all schemata in the 5 million tables corpus introduced in Section 2. We consider the ordered set of column headers and the host part of a web table's URL, which we use to identify web sites, as its schema (so no two schemata from different hosts are counted as equal).

Figure 2 shows the cumulative distribution function (cdf) of the number of schemata per host. The series "Hosts" shows the cumulated percentage of hosts with the number of schemata indicated on the horizontal axis. The series "Web Tables" shows the cumulated percentage of web tables from these hosts. We learn two interesting facts from this distribution: First, 75% of all hosts only use a single schema, but these web sites only contain 20% of all web tables. Second, we see that the majority of the web tables are found on hosts that use a rather small number of different schemata. We find 55% of all web tables on hosts with up to 10 schemata (97% of all hosts) and 84% of all web tables on hosts with up to 100 schemata (99.7% of all hosts).

Figure 3 shows the cdf of the number of web tables per schema. The series "schemata" shows the cumulated percentage of schemata which are re-used as many times as indicated on the horizontal axis. The series "Web Tables" shows the cumulated percentage of web tables that use these schemata. We find that 63.7% of all schemata are only used once, but they do not contribute much (ca. 2.5%) to the total amount of tables in the corpus. Considering all schemata which are re-used up to 100 times (98.7% of all schemata) only cumulates to 17% of all tables. The remaining 83% of all tables have a schema that is used by at least 100 other tables on the same web site.

These results support the hypothesis that most web tables in our corpus are generated by content management or related systems. The majority of web tables is found on hosts

that use a relatively small number of different schemata and many schemata are re-used by a larger number of web tables.

As a counter example, the host "en.wikipedia.org", which clearly violates our assumption, has 13 799 schemata in our dataset, which are on average used by 3 web tables. To understand the reasons for such a large number of different schemata on a single web site, we manually inspect the 20 hosts with the highest diversity of schemata. We find that 20% of the hosts actually provide vertical tables (although detected as horizontal tables) and hence a data value was interpreted as column header. 15% have no headers, but the first data row was mistaken as schema by the header detection method, and 25% use data values in the attribute names (i.e., names of sports teams or the date). Only 40% of the tables are manually created tables (the tables were not systematically created from a database).

4. UNION TABLES TO KB MATCHING

Based on the findings from the previous section, we know that a large number of web tables has a schema that is often re-used on the same host. We can use this observation to stitch all web tables that have the same schema into a single, larger table. Each web table can be considered as a view $v_i \in V$ on a larger table t that applies a selection operation $v_i = \sigma_i(t)$ to that table. As proposed by Ling et al. [22], we can stitch these web tables to a larger table, by creating the union $t' = \bigcup_{v_i \in V} v_i$. We will call the resulting larger tables "union tables" from now on.

The original work of Ling et al. describes how web tables from the same web site can be stitched in order to create larger tables that are useful for visualising and mining the data. The authors define web tables to be stitchable if all their column headers match and create the union of all such tables from the same web site. However, the main focus of their work is to extract additional attributes from the web pages (i.e., from the URL, title, text, navigation) on which the web tables were found. Our work has a different focus and investigates the effect of this procedure on the web table matching task.

The result of creating the union tables for all web sites in our corpus is that the total number of tables is reduced from 5 176 160 to 261 215. The average size of the tables increases from 9 rows to 108 rows for the union tables. By removing exact duplicates from the union tables, we remove on average 8% of the rows. For the 401 hosts in our sample, which contain a total of 3.5 million web tables, we create 16 367 union tables.

To measure the improvement for our initial task, we run T2K Match and COMA on the union tables. We transfer each correspondence created for a column in a union table to all original columns in the web tables that were combined into this union table. We then evaluate the correspondences for the same sample of tables as in Section 2.

On the original web tables, T2K Match achieved an F1-measure of 0.24. With the union tables, this is improved to an F1-measure of 0.5. We see an increase in the number of correct "rdfs:label" correspondences from 95 to 169. For all correspondences, a precision of 0.4 and a recall of 0.66 are achieved. The reason for the still rather low precision is that in total 727 tables were matched to the knowledge base, although only 204 of the sampled tables can actually be mapped.

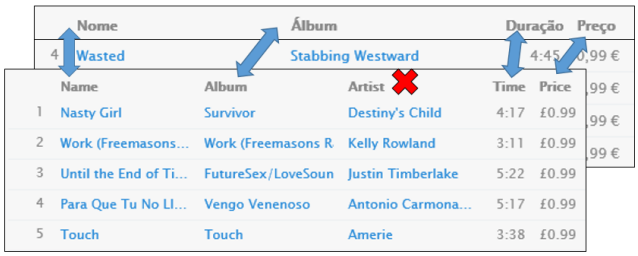


Figure 4: Example of tables that are missed by the union approach.

For COMA, we have to sample data from union tables that exceed 1 000 values per attribute. We first sample all rows that are in the original web tables that are evaluated and then pick rows at random from the union table until 1 000 rows are reached. COMA achieved an F1-measure of 0.37 on the original web tables, which is increased to 0.47 for the union tables. The number of correct correspondences to "rdfs:label" is increased from 18 to 62 and overall a precision of 0.57 and a recall of 0.4 are achieved.

This experiment shows that the creation of union tables improves the result of the applied matching methods. In addition to a larger number of matched tables, we also see that more properties can be matched correctly using the enlarged set of property values in the union tables. However, the achieved performance is still lower than the performance achieved on the gold standards, which is why we test if stitching the union tables into larger tables results in further improvements.

5. STITCHING UNION TABLES

Creating union tables already strongly improved the matching performance compared to the result on the original web tables. Now, we are interested if stitching union tables with overlapping schemata can result in a further improvement. In this section, we first show an experimental evaluation of different schema matching methods on the task of matching union tables. Then, we explain how we combine these methods with ideas from holistic schema matching into a hybrid matcher for our task. Using this matcher, we find mappings between the union tables and use these mappings to stitch the union tables into larger tables.

Examples of tables that cannot be stitched using the union approach are tables which either have the same schema but different column headers (for example, different languages or additional values in the column headers) or which have different, but overlapping schemata (for example a different set of attributes for the same entity). Figure 4 shows an example of two tables with stitchable schemata, which differ in the used language and used the set of attributes.

5.1 Standard Matchers

The schema matching problem for union tables is different from the general web table matching problem, because we can assume that all data was generated from the same underlying table or database. This reduces heterogeneity and we can use exact value comparisons instead of similarity measures for comparing attribute values. To determine the performance of standard matching methods for this task,

Table 1: Datasets for schema matching of union tables.

Dataset	Tables	Rows	Columns	Union	Topic	Characteristics
data.bls.gov	10 825	54 196	59 093	10	statistics	various numeric attributes non-mappable
itunes.apple.com	42 729	494 302	258 275	36	music	different languages, similar attributes
websitelooker.com	11 351	99 865	39 535	4	statistics	similar, but non-mappable attributes
www.nndb.com	23 522	231 738	116 716	9	multiple	attributes with similar domains
seatgeek.com	157 581	2 644 035	630 063	64	multiple	mixture of venues, music and sports
vgchartz.com	23 258	58 637	116 285	6	video games	only one entity per table
Σ	269 266	3 582 773	1 219 967	129		

we experiment with three matching approaches: label-based, value-based and duplicate-based schema matching.

Label-based Matcher. We applied label-based matching implicitly when creating the union tables: all columns with the same header are matched to each other. We can expect that this also works for different union tables. However, if different synonyms or languages are used or if column headers are missing, this approach will miss correspondences.

Value-based Matcher. In cases where column headers are missing or uninformative, it is reasonable to look at the values of the attributes. If two columns contain the same values, they might represent the same attribute. We do not have to consider value similarity, so we only need to measure the value overlap between two columns. However, the weakness of this approach is that if two tables have different, but very similar attributes, for example "birth date" and "founded date", it can be difficult to distinguish correct from coincidental matches.

Duplicate-based Matcher. If value-based similarities are misleading, duplicate-based schema matching is more promising. Given a set of duplicate records, this approach only compares the values of these duplicates in order to find schema correspondences [5]. For each pair of duplicate records, all values are compared, resulting in an attribute similarity matrix for each duplicate. These matrices are then aggregated by averaging, leading to a final attribute similarity matrix. With this approach, attributes with similar domains can be differentiated more precisely, as only attribute values from records describing the same real-world entity are compared.

In order to come up with a suitable set of duplicates, we experiment with three different strategies: we either use candidate keys, determinants, or entity labels to estimate whether two records refer to the same real-world entity. Candidate keys uniquely identify a record and should hence result in an optimal set of duplicates. However, our experiments show that relying on candidate keys does not result in many duplicates. Determinants are smaller sets of attributes than candidate keys and the combination of their values can appear for multiple records in the same table, which results in more duplicates. We use HyFD [25] to determine functional dependencies and candidate keys. Entity label columns are commonly used in web table to knowledge base matching [21, 24, 28, 31, 35]. These columns contain the names of the entities that are described in a table and act as pseudo keys, resulting in the largest number of duplicates.

5.2 Datasets

To evaluate the proposed matching methods, we select several web sites with different characteristics and create union tables from the tables on these web sites. Table 1 gives an overview of the datasets that we use to compare the matching methods. Each dataset contains all web tables from a certain web site (identified by the host part of its URL) that are included in the 5 million table corpus introduced in Section 2. The first three columns in Table 1 show how many web tables, rows and columns each dataset contains. The column "Union" shows the number of schemata, which is also the number of union tables that were created for the respective web site.

5.3 Evaluation

The goal of our evaluation is to measure how correct and complete the result of a matching algorithm is. For this purpose, we manually generate reference mappings for all datasets. These reference mappings are complete, i.e., they contain all correct correspondences between the union tables of the respective datasets. Given the reference mapping M_{ref} and the matcher output M_{match}, we define precision as $P = \frac{|M_{match} \cap M_{ref}|}{|M_{match}|}$ and recall as $R = \frac{|M_{match} \cap M_{ref}|}{|M_{ref}|}$.

5.4 Standard Matcher Experiments

We now present the results of the schema matching experiments for all matchers described in Section 5.1. An overview of all results is shown in Table 2.

Label-based Matcher. The label-based method is very strong in our use case and is overall the best standard matcher with an average F1-measure of 0.69. It is important to note that it achieves a precision of 100% for all datasets.

Value-based Matcher. The value-based method results in the second highest average F1-measure with 0.53. However, its precision is the lowest with an average of only 0.45. The problem is that seemingly similar columns are mapped to each other, which actually represent different attributes. For example, this results in mappings between "artist" and "album" for itunes, "text" and "primary country" for websitelooker, "employment per thousand jobs" and "percent of state employment" for bls and between "founded" and "birth date" for nndb.

Duplicate-based Matcher. The three approaches for duplicate-based matching achieve the lowest average F1-measure values with 0.26 for entity-labels, 0.35 for determinants and 0.11 for candidate keys. However, these methods can only create correspondences if two tables contain records that are duplicates, so all correspondences between tables with distinct records are missed. This is also reflected in the results, as the precision of the duplicate-based meth-

Table 2: Comparison of all schema matchers (F1-measure, best configuration for each dataset is shown in boldface).

	Standard Matchers					Hybrid Matchers				
	Value	Entity Label	Determinant	Key	Label	Value	Entity Label	Determinant	Key	Label
bls	0.67	0.38	0.14	0.03	**0.75**	**0.89**	0.85	0.84	0.84	0.75
itunes	**0.61**	0.45	0.51	0.15	0.18	**0.81**	0.71	0.79	0.53	0.18
websitelooker	0.43	0.20	0.25	0.00	**0.73**	**0.80**	0.71	0.73	0.73	0.73
nndb	0.38	0.10	0.38	0.11	**1.00**	0.52	0.87	**1.00**	**1.00**	**1.00**
seatgeek	0.20	0.27	0.46	0.17	**0.83**	0.78	0.73	**0.87**	0.84	0.83
vgchartz	**0.88**	0.19	0.35	0.19	0.63	**0.96**	0.70	0.70	0.63	0.63
Average	0.53	0.26	0.35	0.11	**0.69**	0.79	0.76	**0.82**	0.76	0.69

ods is much higher with averages of 0.65 for entity labels, 0.89 for determinants and 0.78 for candidate keys. Using entity labels to find duplicates, the problem is that ambiguous names cause incorrect duplicates, which in turn reduces the schema matching performance. This problem is reduced when using determinants or candidate keys. However, we then find fewer duplicates which means that some union tables cannot be matched or are matched incorrectly due to too few duplicates.

The results of this experiment (Table 2, Section *Standard Matchers*) show that the standard matchers cannot satisfactory solve the task of matching union tables individually.

5.5 Hybrid Matcher

The experiments in the previous section have shown that the standard matchers cannot satisfactory match the union tables individually. But, as the value- and duplicate-based matchers use different signals for their matching decisions than the label-based matcher, we can combine them into a hybrid matcher. For this hybrid matcher, we also include ideas from holistic schema matching that make use of the attribute labels, which are the strongest signal in our use case. Holistic schema matching refers to methods that make decisions based on observations from all schemata in a certain corpus [14, 15, 30, 34].

Overall Process. Traditional matching methods only consider two tables at a time. But, if many tables have to be matched, pair-wise matching can produce inconsistencies. For example, two columns from the same table could be matched to each other via a path of correspondences to other tables. Such inconsistencies can only be detected when considering the mapping of more than two tables at the same time. Our hybrid matcher first uses one of the standard matchers from the previous section to perform a pair-wise matching between all union tables. The runtime of the pair-wise matching is quadratic in the number of tables, but as we match the already created union tables and not to the original web tables, this number is rather low in most cases (see Table 1, Column *Union*). After the pair-wise matching, two holistic refinement steps are applied to remove inconsistent correspondences and add missing correspondences to the mapping.

Pair-wise Refinement. The first refinement operates on pairs of tables and the schema correspondences that were created by the standard matcher. First, correspondences which are inconsistent w.r.t. all schemata on the same web site are removed as follows: We assume that attributes are not duplicated in a single schema and use co-occurrence of attribute names in the same schema as negative evidence. This has been shown to be applicable for web tables [16] as well as the schemata of query forms on web pages [14]. All column headers that appear in the same schema cannot be mapped to one another and such correspondences are removed. [7] To exclude violations of this assumption, we filter out all "horizontally stacked" tables as described in [18]. Such tables are constructs where one schema is repeated multiple times in a single table to stretch it horizontally (for purely visual reasons). Second, correspondences for all columns with equal headers are added if they do not exist already, which equals to running the label-based matcher.

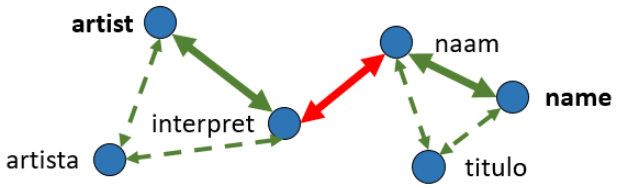


Figure 5: Graph refinement: If "artist" and "name" appeared in the same table, the path between these nodes (solid edges) can be detected as inconsistent. To solve this inconsistency, we remove the red edge, which has the highest betweenness centrality of all edges on this path.

Graph-based Refinement. The second refinement operates on the graph of all schema correspondences as edges and columns as nodes. Again, the first step is to detect and remove inconsistencies. We apply the same rule as during the pair-wise refinement, though this time to the full graph. The correspondences in the graph are inconsistent if two columns with headers that co-occurred in a schema are connected by a path of schema correspondences. In such a case, we remove the edge with the highest betweenness centrality from the inconsistent path. As an example, assume a perfect mapping between columns that represent two attributes. The resulting graph contains two components of connected columns, one for each attribute. Now we add an incorrect correspondence to this graph, which connects the two components, as shown as the red edge in Figure 5. This new edge is part of all shortest paths that connect any two columns from the different components. So, in an inconsistent matching graph, we want to remove the edge(s) that connect different components, which can be measured using betweenness centrality. The betweenness centrality

[7] But if an attribute name occurs multiple times in a single schema, this does not lead to the removal of any correspondences.

measures which fraction of all shortest paths between any two nodes in the graph include a certain edge. In our example, the inserted edge has the highest betweenness centrality value because it is the only connection between the two components. This procedure is similar to the one proposed by Wang et al. [33]. As final step of the graph-based refinement, we add all edges that can be inferred by transitivity to the graph and thus reduce the incompleteness of the graph.

5.6 Hybrid Matcher Experiments

In our next experiment, we run the hybrid matcher on the datasets introduced in Section 5.2 in configurations with all standard matchers discussed in Section 5.1.

Table 2 shows the F1-measure for all standard and hybrid matcher configurations (the best configuration for each dataset is shown in boldface). The best result over all datasets is achieved by the determinant matcher, with an average F1-measure of 0.82. For all methods, a large improvement can be attributed to the correspondences from the label-based matcher, because it can find the matches between tables with disjoint values. In combination with the transitivity applied by the Graph-based Refinement, the value- or duplicate-based and the label-based correspondences complement each other. For example for the itunes dataset with the determinant-based matcher, this results in a recall of 0.69, which exceeds the combined recall of label-based (0.1) and determinant-based (0.39) alone. Finally, the removal of inconsistent correspondences leads to an improvement in precision for all configurations and datasets and does not cause a decrease in recall.

These results show that the combination of the different standard matchers into a holistic hybrid matcher improves the schema matching performance. Depending on which standard matcher is used, we can observe a trade-off between precision and recall. While the value-based matcher still achieves the highest recall at comparably low precision, the duplicate-based methods achieve a higher precision.

5.7 Matching Runtime

As some of our matchers rely on the comparisons of duplicates and as the number of duplicates varies strongly between the different strategies, we now discuss the runtime of the different matcher configurations.

Figure 6 shows the runtime of the hybrid matcher in milliseconds. All experiments were executed on a machine with two Intel Xeon CPU E5-2640 (2.60GHz, Octa-Core), 386GB main memory running Debian release 3.16. The runtime for the label-based method remains below one second for all datasets, as only a very small number of attribute names is compared. The value-based method runs between one and ten seconds for all datasets. As we do not need similarity measures, the value-based matching can be performed by checking the overlap of the domains of the different attributes. However, the higher runtimes are caused by cases where too many correspondences are generated. This leads to a high number of nodes n and edges e which all have to be considered in the graph-based refinement step (the calculation of the betweenness centrality has a runtime in $O(n^2 \log n + ne)$). For the duplicate-based methods, we first generate duplicates from all rows in the tables and then check the attribute values of these duplicates. Hence the runtime is mainly determined by the number of duplicates that is found. Entity label matching, which finds the largest number of duplicates, also has the highest runtime for all datasets (more than 13 minutes for the websitelooker dataset). Determinant matching has the next-highest runtimes, but even the longest running dataset (itunes with 40 seconds) is much faster than the entity label approach. The reason is the much smaller number of duplicates that need to be checked. The runtime of the candidate key matching is close to the value-based method, which is because only very few duplicates are found.

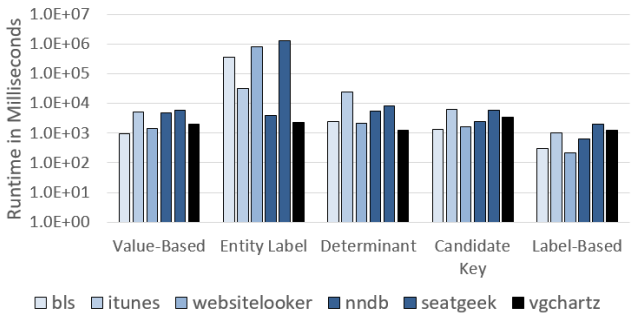


Figure 6: Hybrid matcher runtime.

6. STITCHED UNION TABLES TO KB MATCHING

In the previous section, we have evaluated different methods for matching the union tables, that were created from the initial web tables, to each other. Now, we stitch the union tables using the obtained mapping and evaluate how this changes the schema matching performance for the knowledge base augmentation task. The union tables $u_i \in U$ can be considered as different projections π_i of the same underlying table $u_i = \pi_i(t)$. Using the mapping M obtained by the schema matching, they can be stitched together as $t'' = \bigcup_{u_i \in U} \pi^{-1}(u_i, M)$, where $\pi^{-1}(u_i, M)$ is the transformation of u_i into the consolidated schema for M.

In general, we want to combine as many union tables as possible, but at the same time avoid too much de-normalisation, as the algorithm using the stitched union tables might not be able to deal with highly de-normalised tables. Hence, we decide to merge all union tables that have matching candidate keys, which contain at least one string column. [8] All correspondences between tables that do not fulfil this requirement are ignored during stitching. Using the remaining correspondences, we merge all columns which are connected via schema correspondences into the same column in the stitched union table. The consolidated schema of a stitched union table consists of all merged columns and all columns without correspondences. After transforming every union table into its respective consolidated schema, we create their union to obtain the final, stitched union tables. To avoid extremely sparse columns in these tables, all columns with more than 95% empty values are removed.

6.1 Matching Individual Web Sites

To test the influence of the matching method for stitching union tables on the performance of T2K Match, we run experiments with the different configurations of our hybrid

[8] T2K Match relies on entity label columns for the matching, so all tables without a string column that exceeds the uniqueness threshold are ignored.

Table 3: Datasets for stitched union table to knowledge base matching. T = Tables, C = Columns, Det. = Determinant, CK = Candidate Key.

	Union		Value		Entity		Det.		CK	
Dataset	T	C	T	C	T	C	T	C	T	C
itunes	36	226	1	10	1	12	1	10	1	16
nndb	10	28	7	20	7	23	9	27	9	27
seatgeek	64	233	2	7	2	7	4	15	5	18
vgchartz	6	29	2	17	2	18	3	19	3	19

Table 4: Evaluation of T2K Match for web tables, union and stitched union tables.

	itunes	nndb	seatg.	vgchartz	avg.
Web Tables	0.500	0.515	0.038	0.244	0.324
Union	0.799	0.999	0.799	0.800	0.849
Value-Based	0.913	0.997	0.800	0.500	0.802
Entity Label	0.982	0.998	0.799	0.500	0.820
Determinant	0.944	0.999	0.800	0.800	0.886
Cand. Key	0.897	0.999	0.800	0.800	0.874
Label	0.930	0.999	0.800	0.800	0.882

matcher on the datasets introduced in Section 5.2. A precondition for the schema matching for knowledge base augmentation is that the tables contain both entities and attributes which already exist in the knowledge base. As the websitelooker and bls datasets do not contain any attributes that also exist in the knowledge base, we remove them from this experiment. We apply the hybrid matcher in its different configurations to all datasets shown in Table 3. The different columns show how many tables (T) and columns (C) resulted from the stitching with the different matchers.

Table 4 shows the F1-measure resulting from running T2K Match on the original, union and the stitched union tables for all configurations of the hybrid matcher. The reference mapping for all datasets was created by manually labelling the union tables with their corresponding properties in DBpedia and transferring the correspondences to all original web table columns. We evaluate all correspondences that are created for the original web tables.

For the original tables, T2K Match manages to create rather precise results with low recall for the itunes and nndb datasets but fails for the seatgeek and vgchartz datasets. The average F1-measure is 0.32. The tables in the vgchartz dataset can be extremely small and the result completely depends on whether the one entity in the table can be correctly recognised. For the seatgeek dataset, the described venues are not recognised as main entity, so no schema correspondences can be created.

With the union and stitched union tables, the average F1-measure increases to a range from 0.8 to 0.88. Most of the attributes which exist in the knowledge base have been correctly matched by all stitching approaches, with the result that the performance of T2K Match does not differ much on these datasets. However, we see improvements for the itunes dataset, where the smaller union tables resulted in missed and incorrect correspondences. For these tables, the stitched union tables obtained with the determinant-based matcher resulted in the best performance of T2K Match. For the vgchartz dataset, the results of the value-based and entity-label based stitching approaches cause a worse performance of T2K Match compared to the union tables and the other stitching approaches. This is because union tables which only have a single attribute in common are matched, leading to an incorrect decision of T2K Match. With the other approaches, more attributes are required and hence this error is avoided.

Overall, we can achieve improvements of 0.48 to 0.76 points in F1-measure with the union and stitched union tables in comparison to the original web tables. In all cases, creating the union tables causes a huge increase in F1-measure. Depending on the dataset, the additional stitching step after creating the union tables can further improve the results. However, as not all attributes can be mapped to the knowledge base, the differences between the stitching approaches are only slight in this experiment.

6.2 Matching the Random Sample

We now come back to our initial experiment and run the stitching on all web sites in our random sample of 1 000 tables from Section 2 and afterwards apply T2K Match and COMA. Again, each correspondence for a column in a stitched union table is transferred to all original web table columns that were merged into this column. We then evaluate all correspondences for the web tables in the initial sample. As in Section 4, we first stitch all 3.5 million web tables (16 379 union tables) from the hosts in our sample and then apply T2K Match and COMA. This results in 1 160 stitched union tables for the value-based matcher, 1 562 for the entity-label matcher, 1 981 for the determinant matcher and 2 388 stitched union tables for the candidate-key matcher.

Table 5 gives an overview of the results for the different stitching steps and matcher configurations. For T2K Match, the initial result of 0.24 F1-measure is improved to 0.50 when creating the union tables. With the stitched union tables, the best performance is achieved in the configuration with the determinant matcher, with an F1-measure of 0.62. The additional improvements are due to an improved precision for correspondences to "rdfs:label" and improvements in both precision and recall for other properties.

For COMA, we also see a further improvement in both precision and recall. We use only the determinant-based matcher to create stitched union tables in this experiment. The F1-measure increased from 0.37 for the original web tables and 0.47 for the union tables to 0.51 for the stitched union tables. Although the performance is worse than with T2K Match, which is specialised for this task, we can see an improvement when using union tables (10%) and stitched union tables (14%) instead of the original web tables.

Our experiments show that the step from original web tables to union tables results a strong increase in performance on our sample of 1 000 web tables. We thus recommend any web table matching system to at least implement creating union tables as a preprocessing step, which does not require any schema matching. If the matching results should be further improved, systems can decide to combine union tables into stitched union tables using schema matching.

Choosing a Matcher. The experiments on the selected websites showed that the differences in the performance of our matching approaches are not necessarily reflected when running T2K Match. The reason is that not all of the attributes in these tables have a counterpart in the knowl-

Table 5: Results of running T2K Match and COMA with web tables, union tables and stitched union tables.

T2K Match	Precision	Recall	F1-measure
Web Tables	0.20	0.32	0.24
Union Tables	0.40	0.66	0.50
Value-based	0.58	0.59	0.58
Entity-label	0.56	0.56	0.56
Determinant	0.63	0.61	0.62
Candidate Key	0.62	0.60	0.61
COMA			
Web Tables	0.54	0.28	0.37
Union Tables	0.57	0.40	0.47
Stitched Union	0.62	0.43	0.51

edge base, and are hence not considered in this evaluation. On the random sample, however, the determinant-based matcher achieves the best result, which confirms the finding from Section 5.6. In general, the results show that reasonable choices for a standard matcher are the value-based, the determinant-based and the candidate-key matcher. The entity-label matcher tends to create too many duplicates which heavily impacts the runtime but does not lead to a better result. In the experiments from Section 5.6, the value-based matcher achieves the highest recall and the candidate-key matcher reaches the highest precision. As the determinant-based matcher achieves the highest F1-measure, it is preferred unless a use case requires a focus on either precision or recall. In the experiments on the random sample, it also achieves the highest precision and recall among all matcher configurations for the web tables to knowledge base matching use case.

7. RESULT ANALYSIS

In the previous sections, we have shown that stitching web tables can improve the results for the web table to knowledge base matching task. In this section, we analyse the impact of the stitching procedure on the amount of correctly extracted values and the amount of tables in the corpus. Afterwards, we investigate which characteristics of the set of tables per host lead to the highest performance gains.

7.1 Amount of Extracted Values

With respect to applications that use the data from the tables, such as filling missing values in a knowledge base, it is not only important to correctly match the schema, but rather how many values are extracted correctly. Correctly extracting a value requires not only the schema to be matched correctly, but also the correct identification of the entity that is described in a specific table row. We estimate the precision of the entities identified by T2K Match using a sample of 200 entities for each configuration from our 1 000 tables experiment. Based on the original evaluation of T2K Match, we expect a precision of 0.9 for the row-to-instance matching task [28]. We find that the precision on the original web tables is 98%, and on the union and stitched union tables both 87%. This difference has mainly two reasons. First, most errors are made for entities with ambiguous names. Avoiding these errors requires to exploit multiple attributes per row and to put less weight on the entity label than done by T2K Match. Second, the tables containing these entities are often not mapped using

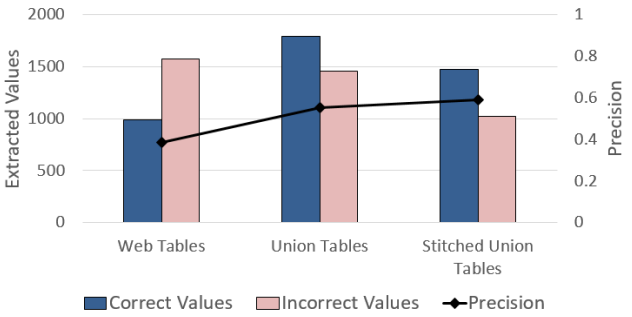


Figure 7: Evaluation of extracted values.

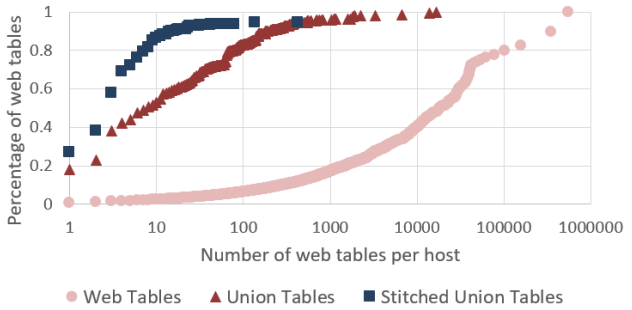


Figure 8: Tables per web site at different stitching steps.

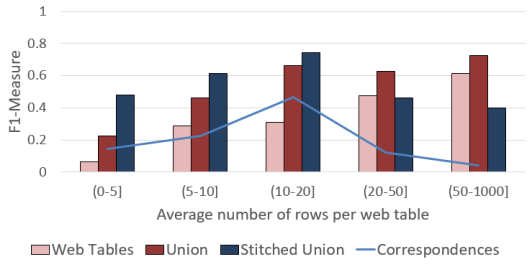
the original web tables, which is why these errors are not made in this case.

Based on the schema and entity mapping, we calculate the amount of values that can be correctly extracted. Note that correctly extracted in this context means that entity and attribute correspondence is correct, not that the actual attribute value is true. Figure 7 shows that by using union tables and stitched union tables, we can extract more values at a higher precision than for the original web tables. In a knowledge base augmentation scenario, the next step would be to apply a data fusion method to the extracted values from all tables in order to determine potentially true values. However, this exceeds the scope of this paper, but was demonstrated earlier for a similar web table corpus [29].

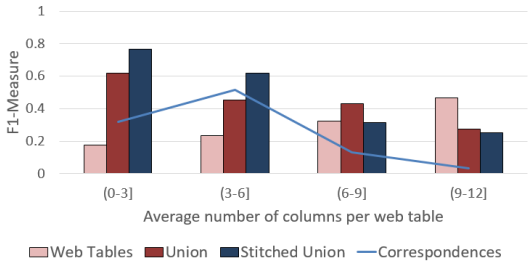
7.2 Table Stitching Statistics

The result of the stitching procedure is the combination of large numbers of small web tables into larger tables. This reduces the overall amount of tables in the corpus as well as the amount of tables for individual web sites. Besides the benefit for the knowledge base augmentation task, a reduced amount of tables can also be beneficial for other applications. For example, table search [7,20,34] or interactive exploration of a table corpus [12] are simplified by a reduced amount of tables. In order to see the impact of table stitching on such applications, we measure the number of tables per host at each stitching step for the web tables corpus introduced in Section 2, shown as cdf in Figure 8. The horizontal axis shows the number of tables per web site and the vertical axis indicates the cumulated percentage of web tables, union and stitched union tables that are found on these web sites. For generating these statistics, we use the hybrid matcher in its configuration with determinant matching.

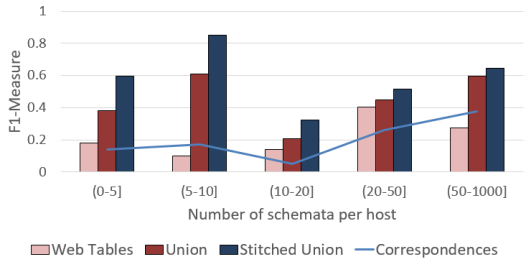
In the original corpus, we find many web sites with large



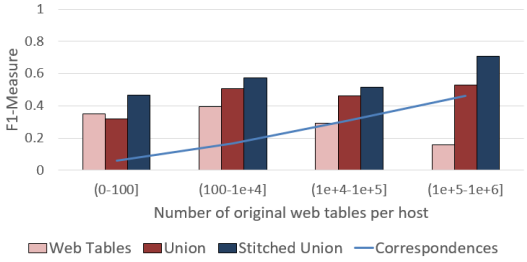
(a) Average number of rows per web table.



(b) Average number of columns per web table.



(c) Number of schemata per host.



(d) Number of original web tables per host.

Figure 9: Influence of table characteristics on matching performance.

amounts of tables and a total of 5 176 160 web tables. After creating the union tables, the total number of tables is drastically reduced to 261 215. We now find the data of 50% of all original tables on web sites with no more than 7 union tables and reach the 75% mark at 42 union tables per website. Applying the union table stitching to all hosts with up to 1 000 union tables further reduces the amount of tables to 104 221. [9] More than 50% of the original tables now belong to web sites with a maximum of only 3 stitched union tables and we find 75% of the original tables on websites with less than 7 stitched union tables.

7.3 Table Characteristics

Another important aspect is how effective the stitching methods are for hosts that publish web tables with different characteristics. Hence, we analyse for which table characteristics the stitching methods result in the largest benefits. Figure 9 shows the web table to knowledge base schema matching performance w.r.t. four characteristics (average number of rows & columns, total number of schemas & tables) on host level. These characteristics can be determined from the original web tables for each host, so this profile can be used to determine whether a stitching procedure should be applied or not. The bars in Figure 9 show the F1-measure for the random 1 000 table sample, the line indicates the share of all correspondences that falls into each bin. Figure 9a shows that stitching is effective for web tables with few rows (up to 20 rows per web table), which is our main argument for applying a stitching procedure. For higher numbers of rows, the results of using stitched union tables are worse than the results of just using union tables. These are web tables which are already large enough for matching them directly, so every error that is introduced by the union table stitching procedure has a negative impact on the result. Figure 9b shows that a similar trend can be observed for the number of table columns. With up to 6 columns, the stitching approach is effective, but for tables with more columns the performance drops. Figure 9c shows that stitching improves the results regardless of the number schemata per host. However, the advantage over only using the union tables is highest for up to 20 schemas. Figure 9d shows that stitching is an improvement regardless of the number of original web tables provided by a host. We can further see a trend that the improvement increases with the amount of web tables.

[9]We set a threshold of 1 000 union tables per host to exclude web sites that cannot be handled by our method, such as Wikipedia.

8. RELATED WORK

Since the first large-scale effort to extract tables from web pages by Cafarella et al. [8] in 2008, many systems were developed that facilitate web table data. The tasks that these systems address range from data search and table extension [7, 20, 21, 23, 34], over knowledge base completion [11, 28, 31, 32, 35], to displaying web table data as rich snippets in search engine results [2]. The topics that are covered by large web table corpora were analysed by [13] and [29].

Web Table to Knowledge Base Matching. The task of matching the schemata of web tables to a knowledge base is addressed by Limaye et al. [21], Mulwad et al. [24], Venetis et al. [31] and Zhang [35]. The approaches of Limaye, Mulwad and Venetis only consider columns which contain named entities. Zhang also creates correspondences for literal columns. The method of Zhang complements our work, as the focus of the method is the efficiency of the matching process on large tables, which involves sampling the rows in order to reduce the runtime.

Mulwad and Zhang evaluate on the table corpus used by Limaye, but create their own versions of the gold standard. Hence, the achieved performances of 64% accuracy by Limaye, 86% F1-measure by Mulwad and 66% F1-measure by Zhang cannot be directly compared. Venetis et al. evaluate

on a random sample, from which web tables are removed for which their algorithm did not produce any annotations or where the entity label column was not correctly identified. This sample is hence not comparable to our sample, which contains many tables that cannot be mapped to the knowledge base. They note that it was hard to find relations which are supported by enough rows in the tables, which could indicate that their method can also be improved by first stitching the web tables.

Union Tables. All of the mentioned matching approaches consider the extracted web tables as individual data sources. Ling et al. [22] proposed to create the union of web tables with matching headers. But different from our work, their focus is the extraction of context attributes from the text surrounding the tables and to find proper attribute names for these context attributes. To the best of our knowledge, we are the first to apply the idea of table stitching to a large number of web tables in order to improve matching performance. Other research that considers the extraction of context information in order to improve the matching performance includes the work by Zhang [35] and Braunschweig [6] as well as the Octopus [7] and Infogather [34] systems.

Web Table to Web Table Matching. Matching web tables to each other and not to a knowledge base is mainly considered in the context of table extension, where the user provides a query table that is enriched with the data from matching web tables from a large corpus [7, 9, 20, 34].

Das Sarma et al. [9] propose a method with a very similar idea to our method, but a different use case. They propose two operations that find similar tables in a corpus to add additional rows and attributes to a given query table. However, their methods are designed to find such tables for a single query table, while we apply these ideas to combine the web tables in a corpus.

The Octopus [7] system provides similar operations. Given a query table, potentially matching web tables are discovered from a large corpus. Further, it provides operations to extract context attributes from the web pages, create the union of multiple search results and add additional columns from other web tables. In the discussion of their web table matching method, the authors recognise the problem that two related web tables, which could have been created from the same underlying table, might not have any data in common, which prevents a match. They solve this problem by introducing the column width as additional feature. According to their description, we assume that this problem would be solved when stitching the web tables before their matching operation.

The Infogather system by Yakout et al. [34] also provides for extending a base table with additional attributes. This system first matches all web tables in a corpus in a pairwise way and then holistically refines the matches in the graph created by the schema correspondences using variations of PageRank. The main idea is to find tables which have no direct matches with the query table via this matching graph. If such an indirect match is created via the connections among different tables from the same web site, the result is comparable to using direct matching with one of our stitched tables.

Schema Matching Techniques. In our experiments, we have compared different schema matching methods for the task of matching union tables. According to the classification of Rahm and Bernstein [26], these matchers belong to the two major categories of schema-only matchers and instance-based matchers. With our hybrid matcher, we also incorporate holistic correspondence refinement, which belongs to the category of collective matching approaches according to Bernstein et al. [3].

The holistic matching methods in our hybrid matcher have been proposed before by He et al. [16] and He and Chang [14]: Both consider co-occurrence of attribute names in a schema as negative evidence. He and Chang [14] propose a method for matching web query interfaces. They consider attributes that occur in the same schema as non-matches and all other attribute combinations as potential matches.

He et al. [16] focus on synonym discovery and use web tables in combination with search query logs. Different users, which search for the same attribute, likely enter different synonyms as query terms, but click on the same results returned from a search engine. This signal is used as positive evidence for attribute synonymity. Attributes which co-occur in the same web table, however, are likely not synonyms, which is why co-occurrence is considered as negative evidence.

The removal of inconsistent correspondences in our graph-based refinement step is similar to the cross validation of mappings proposed by Wang et al. [33]. They create a graph of schema elements with correspondences between them. In addition to correspondences between schema elements, they use a predefined global schema to define clusters in this graph. To remove incorrect correspondences, they move schema elements between the clusters until the edge-cut, i.e., the sum of weights of the edges between different clusters, is minimised. Due to the lack of a global schema to determine such clusters, we remove the edges with the highest betweenness centrality in our approach.

9. CONCLUSION

The research area of web table to knowledge base matching is becoming increasingly sophisticated and matching systems are going long ways for small performance improvements on the common benchmarks (see Ritze and Bizer [27] for a recent overview). This paper has shown that table stitching has a strong impact on the matching quality, but up till now, none of the existing web table to knowledge base matching systems uses table stitching as a pre-processing step. Thus, the conclusion of this paper is that web table matching systems should combine web tables into union tables and for further performance improvements combine union tables into stitched union tables in order to be able to properly exploit the large number of very small tables that exist on the Web.

An interesting direction for future work is the matching of stitched tables across different web sites. As table stitching results in larger tables and reduces the overall number of tables in a corpus, it appears more feasible to deal with the increased heterogeneity and the higher runtime requirements of the cross-site table matching scenario. The resulting data sets would not be limited to existing entities or properties in a knowledge base and could thus be used for the construction of more comprehensive knowledge bases that cover long-tail entities and properties.

10. REFERENCES

[1] D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with coma++. In *Proc. of the 2005 SIGMOD*, pages 906–908, 2005.

[2] S. Balakrishnan, A. Y. Halevy, B. Harb, and et al. Applying webtables in practice. In *Conference on Innovative Data Systems Research CIDR*, 2015.

[3] P. A. Bernstein, J. Madhavan, and E. Rahm. Generic schema matching, ten years later. *PVLDB*, 4(11):pages 695–701, 2011.

[4] C. S. Bhagavatula, T. Noraset, and D. Downey. Tabel: entity linking in web tables. In *International Semantic Web Conference*, pages 425–441. Springer, 2015.

[5] A. Bilke and F. Naumann. Schema matching using duplicates. In *21st ICDE*, pages 69–80. IEEE, 2005.

[6] K. Braunschweig, M. Thiele, J. Eberius, and W. Lehner. Column-specific Context Extraction for Web Tables. In *Proc. of the 30th ACM Symposium on Applied Computing*, SAC '15, pages 1072–1077, 2015.

[7] M. J. Cafarella, A. Halevy, and N. Khoussainova. Data Integration for the Relational Web. *PVLDB*, 2(1):1090–1101, 2009.

[8] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. WebTables: Exploring the Power of Tables on the Web. *PVLDB*, 1(1):538–549, 2008.

[9] A. Das Sarma, L. Fang, N. Gupta, A. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu. Finding Related Tables. In *Proc. of the 2012 SIGMOD*, pages 817–828, 2012.

[10] H.-H. Do and E. Rahm. Coma: a system for flexible combination of schema matching approaches. In *Proc. of the 28th VLDB*, pages 610–621, 2002.

[11] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion. In *Proc. of the 20th SIGKDD*, pages 601–610, 2014.

[12] J. Ellis, A. Fokoue, O. Hassanzadeh, A. Kementsietsidis, K. Srinivas, and M. J. Ward. Exploring Big Data with Helix: Finding Needles in a Big Haystack. *SIGMOD Rec.*, 43(4):43–54, Feb. 2015.

[13] O. Hassanzadeh, M. J. Ward, M. Rodriguez-Muro, and K. Srinivas. Understanding a large corpus of web tables through matching with knowledge bases: an empirical study. In *Proc. of the 10th Int. Workshop on Ontology Matching*, pages 25–34, 2015.

[14] B. He and K. C.-C. Chang. Statistical schema matching across web query interfaces. In *Proceedings of the 2003 SIGMOD*, pages 217–228, 2003.

[15] B. He, K. C.-C. Chang, and J. Han. Discovering complex matchings across web query interfaces: a correlation mining approach. In *Proceedings of the tenth ACM SIGKDD*, pages 148–157, 2004.

[16] Y. He, K. Chakrabarti, T. Cheng, and T. Tylenda. Automatic discovery of attribute synonyms using query logs and table corpora. In *Proceedings of the 25th WWW*, pages 1429–1439, 2016.

[17] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 6(2):pages 167–195, 2015.

[18] O. Lehmberg and C. Bizer. Web table column categorisation and profiling. In *Proceedings of the 19th WebDB*, page 4. ACM, 2016.

[19] O. Lehmberg, D. Ritze, R. Meusel, and C. Bizer. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th WWW*, pages 75–76, 2016.

[20] O. Lehmberg, D. Ritze, P. Ristoski, R. Meusel, H. Paulheim, and C. Bizer. The Mannheim Search Join Engine. *Web Semantics: Science, Services and Agents on the World Wide Web*, 35:159–166, 2015.

[21] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and Searching Web Tables Using Entities, Types and Relationships. *PVLDB*, 3(1-2):1338–1347, 2010.

[22] X. Ling, A. Y. Halevy, F. Wu, and C. Yu. Synthesizing union tables from the web. In *IJCAI*, page 2677, 2013.

[23] J. Morcos, Z. Abedjan, I. F. Ilyas, M. Ouzzani, P. Papotti, and M. Stonebraker. Dataxformer: An interactive data transformation tool. In *Proceedings of the 2015 SIGMOD*, pages 883–888. ACM, 2015.

[24] V. Mulwad, T. Finin, and A. Joshi. Semantic message passing for generating linked data from tables. In *ISWC*, pages 363–378. Springer, 2013.

[25] T. Papenbrock and F. Naumann. A hybrid approach to functional dependency discovery. *Proceedings of the 2016 SIGMOD*, pages 821–833, 2016.

[26] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):pages 334–350, 2001.

[27] D. Ritze and C. Bizer. Matching Web Tables To DBpedia - A Feature Utility Study. In *Proceedings of the 20th EDBT*, pages 210–221, 2017.

[28] D. Ritze, O. Lehmberg, and C. Bizer. Matching HTML Tables to DBpedia. In *Proc. of the 5th WIMS*, page 10, 2015.

[29] D. Ritze, O. Lehmberg, Y. Oulabi, and C. Bizer. Profiling the potential of web tables for augmenting cross-domain knowledge bases. In *Proceedings of the 25th WWW*, pages 251–261, 2016.

[30] W. Su, J. Wang, and F. Lochovsky. Holistic Schema Matching for Web Query Interfaces. In *EDBT 2006*, volume 3896, pages 77–94. Springer, 2006.

[31] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering Semantics of Tables on the Web. *PVLDB*, 4(9):528–538, 2011.

[32] J. Wang, H. Wang, Z. Wang, and K. Q. Zhu. Understanding Tables on the Web. In *Proc. of the 31st ER*, pages 141–155, 2012.

[33] J. Wang, J.-R. Wen, F. Lochovsky, and W.-Y. Ma. Instance-based schema matching for web databases by domain-specific query probing. In *PVLDB*, volume 30, pages 408–419, 2004.

[34] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. InfoGather: Entity Augmentation and Attribute Discovery by Holistic Matching with Web Tables. In *Proc. of the 2012 SIGMOD*, pages 97–108, 2012.

[35] Z. Zhang. Effective and efficient semantic table interpretation using tableminer+. *Semantic Web*, 8(6):1–39, 2017.