Katrin Braunschweig, Maik Thiele, Julian Eberius, Wolfgang Lehner

Column-specific Context Extraction for Web Tables

Erstveröffentlichung in / First published in:

SAC 2015: Symposium on Applied Computing, Salamanca 13.-17.04.2015. ACM Digital Library, S. 1072–1077. ISBN 978-1-4503-3196-8.

DOI: https://doi.org/10.1145/2695664.2695794

Final edited form was published in "SAC 2015: Symposium on Applied Computing. Salamanca 2015", S. 1072–1077, ISBN 978-1-4503-3196-8
https://doi.org/10.1145/2695664.2695794

Column-specific Context Extraction for Web Tables

Katrin Braunschweig, Maik Thiele, Julian Eberius, Wolfgang Lehner
Technische Universität Dresden
Dresden, Germany
firstname.lastname@tu-dresden.de

ABSTRACT

Relational Web tables have become an important resource for applications such as factual search and entity augmentation. A major challenge for an automatic identification of relevant tables on the Web is the fact that many of these tables have missing or non-informative column labels.
Research has focused largely on recovering the meaning of columns by inferring class labels from the instances using external knowledge bases. The table context, which often contains additional information on the table's content, is frequently considered as an indicator for the general content of a table, but not as a source for column-specific details.
In this paper, we propose a novel approach to identify and extract column-specific information from the context of Web tables. In our extraction framework, we consider different techniques to extract directly as well as indirectly related phrases. We perform a number of experiments on Web tables extracted from Wikipedia. The results show that column-specific information extracted using our simple heuristic significantly boost precision and recall for table and column search.

Categories and Subject Descriptors

H.3.1 [**Information Storage And Retrieval**]: Content Analysis and Indexing

General Terms

Algorithms, Experimentation

Keywords

Information Extraction

1. INTRODUCTION

In recent years, researchers have recognized relational tables on the Web as an important source of information. A wide number of applications, including factual search [14], entity augmentation [2] and ontology enrichment [6], benefit from the vast amount of structured data found in these tables. The recently published Web Data Commons - Web Tables Corpus [1], for instance, contains 147 million relational tables extracted from the Web.

DOI: http://dx.doi.org/10.1145/2695664.2695794

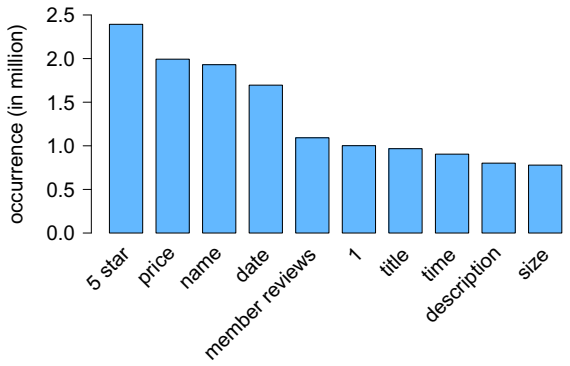


Figure 1: Top 10 most frequent column labels in English subset of WDC table corpus.

These tables vary greatly regarding quality, completeness and consistency of the data. Especially the heterogeneity in the column labels proves a major challenge for the integration of Web tables. In order to combine data from multiple Web tables or integrate it in a local database, we need to understand the content of the table and the intention of the original creator. However, column headers in Web tables are often too ambiguous or too general to support the user (or a search engine) in understanding the column content. Figure 1 shows the distribution of the 10 most frequent labels in the English language section of the Web Tables Corpus. We can see that generic labels such as *name* and *title* are very common. Even worse, a large number of columns, 13% in this section of the corpus, provide no label at all.
To address this challenge, a lot of research effort has been put into inferring better labels from the column content using, for example, external knowledge bases [3, 11, 12]. Yet, in many cases, the main content of the table is mentioned in the text surrounding the table on the Web page. So far, this context has mainly been used as a general indicator for a table's content, but not to identify the content of individual columns. In this paper, our goal is to extract column-specific information from the context in order to improve the following tasks: (1) table ranking and (2) column mapping. Column-specific context can improve the ranking of tables for a query, by giving more value to context that can be linked to a column in the table. Similarly, mapping individual columns to query terms can be improved by providing a richer description for the column.

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

Final edited form was published in "SAC 2015: Symposium on Applied Computing. Salamanca 2015", S. 1072–1077, ISBN 978-1-4503-3196-8
https://doi.org/10.1145/2695664.2695794

In detail, our contributions in this paper are: (1) We propose a heuristic approach to identify column-specific information from the context of Web tables. (2) We design a set of algorithms to extract both, directly and indirectly related phrases. (3) We present experiments on a corpus of real Web tables that show that our heuristic is effective in extracting related phrases that significantly improve table and column search.
The remaining paper is organized as follows: In Section 2, we give a general description of our approach. A detailed description of the different extraction algorithms is given in Sections 3 and 4. An experimental evaluation of our approach is presented in Section 5, followed by related work and a conclusion in Sections 6 and 7, respectively.

2. CONTEXT AND LABEL EXTRACTION

As stated before, our goal is to extract text snippets from the context of a Web table that describe individual columns in the table and to link these new labels to the columns. In this paper, we consider the following context sources:

- **Caption:** First, we consider the <*caption*> tag, which is used to describe the main content of the table. While this context source has the strongest link to the table, it is optional, which means that many tables may not provide a caption.
- **Headline:** The second source is the headline of the section where the table is located, as it may also describe the main content of the table. We extract the last designated headline (<*h1*>-<*h6*> tags) that appears before the table on the page.
- **Surrounding Text:** In addition to the section headline, we also extract the text surrounding the table in this section. We extract all text from <*p*> and <*li*> tags between the section headline and the first headline that appears after the table.
- **Full Text:** While most information related to a table is found close by, some useful information may also be found elsewhere on the page. So, for evaluation purposes, we also extract the complete text (regardless of specific tags) from the body of the Web page.

For most Web tables, the content of the table is mentioned in one or more of these context sources. However, interpreting all the information in the text and extracting the relevant sections is very challenging and requires a complex linguistic analysis, which is not feasible for very large corpora such as the Web Tables Corpus. Instead, we use a simple heuristic to identify relevant text snippets. This heuristic is motivated by two observations: (1) Most tables that are of interest for fact search or entity augmentation describe entities and their attributes. The majority of these attributes can be (and in most cases are) labeled using a noun or noun phrase. (2) The column labels are often very unspecific, while the more specific information is given in the context of the table.
So instead of interpreting the entire context and linking the extracted information to the columns, we go in the opposite direction. Starting with columns that have a noun or noun phrase as a label, we search the context for longer (and thus more specific) noun phrases that are related to these labels. It is clear that this simple heuristic will not be able to identify all related information in the context, as the information can be provided in grammatical structures other than noun phrases. However, our evaluation results suggest that this heuristic is sufficient to improve search results for many columns and tables.

For the extraction of related noun phrases, we distinguish between directly and indirectly related phrases. In the following two sections we describe this distinction in more detail and present the respective extraction approaches.

3. DIRECTLY RELATED CONTEXT

We call a noun phrase in the context *directly related* to a column, if the terms used in the original column label appear in the phrase. For instance, in Figure 2a, the phrase *newly industrialized country* is directly related to the label *country*. We consider two types of directly related context: (1) qualifying noun phrases, i.e. noun phrases that give a more detailed description than the original column label, and (2) phrases that are expansions of acronyms or abbreviations found in the column label.

3.1 Qualifying Noun Phrases

Our goal is to extract noun phrases from the context, that give a detailed description of the content of a column. We call these noun phrases *qualifying* noun phrases. To identify these phrases, we first parse the context and extract all noun phrases and their respective head nouns. We then look for mentions of the column label in these noun phrases. To account for different surface forms of words in the label and the phrases, we lemmatize both before matching.
We follow a conservative strategy when matching the strings, requiring for the original label to match the head of a noun phrase. While this approach might miss some interesting phrases, we found that a more generous strategy introduces too much noise.
After selecting all potentially related noun phrases, in a final step, we apply a filter to remove some incorrect phrases. A common issue we encountered is the mention of column instances (i.e. values in the column) in the context. Consider the following example: our column in question contains city names such as *New York City*, *London* and *Berlin* and has the column label *city*. Following our extraction approach, a mention of *New York City* in the context would be considered a related phrase, since the label is part of the phrase. To address this issue, we filter noun phrases that match instances of the column from the result set. Finally, we link all extracted phrases to the column.

3.2 Acronyms and Abbreviations

Acronyms and other forms of abbreviations are also often found in column labels, whether it is to meet the space limitations of the Web page or simply out of convenience. Unless the abbreviation is common enough to not require an explanation (e.g. abbreviated units such as *km* or *mph*), the correct expansion is often mentioned somewhere in the context. Figure 2b shows such an example, where an acronym, *HDI*, is used as a column label and the expansion, *Human Development Index*, is mentioned in the context. Expansions of acronyms and abbreviations are also relevant phrases that describe the content of a column and can be identified using regular expressions as described Sorrentino et al. in [9]. In this paper, we limit the expansion of short forms to acronyms.
We start the extraction of acronym expansions by first identifying candidate phrases that match the following pattern, where *NP* stands for noun phrase:

$$NP(NP)$$

If the phrase represents the description of an acronym, then the first noun phrase usually contains the expansion, while the noun phrase inside the parenthesis represents the acronym. However, other phrases may also match this generic pattern. So, in a second step, we check if the first noun phrase is indeed an expansion of the second noun phrase by using a regular expression. If multiple noun phrases match the same acronym, we select the shortest (in number of words) and link it to the column.

Final edited form was published in "SAC 2015: Symposium on Applied Computing. Salamanca 2015", S. 1072–1077, ISBN 978-1-4503-3196-8
https://doi.org/10.1145/2695664.2695794

Newly industrialized country
The category of newly industrialized country (NIC) is a socioeconomic classification applied to several countries around the world by political scientists and economists...
Current NICs
The following table presents the list of countries consistently considered NICs by different authors ...

Region	Country	GDP (PPP)	...
Africa	South Africa	555.340	...
North America	Mexico	1,659.016	...
South America	Brazil	2,309.138	...
...	...	...	...

(a)

List of countries by Human Development Index
...

Methodology
... The Human Development Index (HDI) is a comparative measure of life expectancy, literacy, education, standards of living, and quality of life for countries worldwide...

Rank	Country	HDI
1	Norway	0.955
2	Australia	0.938
3	United States	0.937
...	...	...

(b)

List of French Forts in North America
This is a list of all french forts in New France built by the French government or French Chartered companies in ...

Canada
The french forts in Canada were located west from Alberta to the Atlantic Ocean and as far north as James Bay.

Name	Date constructed	...	Country
Port-la-Joye	1720	...	Canada
Fortress of Louisbourg	1719	...	Canada
Fort Ville-Marie	1642-1688	...	Canada
...	...	...	...

(c)

Figure 2: Example tables

4. INDIRECTLY RELATED CONTEXT

The direct matching approach described in Section 3 achieves good results for some tables, but also has some significant limitations. In order to find relevant phrases in the context, it requires the original column label to be a meaningful description of the column content. However, many tables on the Web contain empty or non-informative headers. Figure 2c gives an example of such a non-informative label in the first column. Without the context, the label *name* does not provide any information on what type of entity is listed in the column. In the English subset of the WDC Web Table Corpus non-informative labels such as *name* and *title* are among the most common labels (see Figure 1), and almost 13% of columns have no label at all. To solve this problem, we need to infer suitable labels from the column instances in cases where the original label does not sufficiently describe the column content.

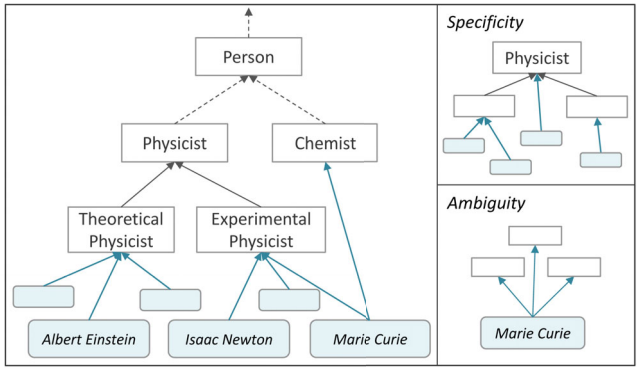


Figure 3: Taxonomy

4.1 Taxonomy Labeling

A frequently applied technique to recreate column labels from instances is to use an external source of commonsense knowledge, such as a taxonomy or ontology. In the literature, different approaches have been proposed to infer suitable column labels from these knowledge bases. A short overview of the related work is given in Section 6. In our work, we use a successive labeling approach. In a first step, we try to match each instance in the column to an instance in the knowledge base in order to retrieve potential type or class labels. In a second step, we aggregate the retrieved labels across the entire column to select the *top k* labels with the highest total scores as suitable labels for the column.
In our approach, we consider a knowledge base that stores two types of relationships between instances and classes: (1) a relationship that expresses that an instance is an example of a class, and (2) a relationship that expresses that a class is a subclass of another class. Figure 3 shows an example of such a taxonomy, where the blue boxes represent instances and white boxes represent classes. To score the classes in the taxonomy, we define two functions: *ambiguity* and *specificity*.

Definition 1. (Ambiguity) This function addresses the fact that the same instance in the taxonomy may have several types, as highlighted in Figure 3. We consider a *type* to be a class that is directly linked to an instance. For example, *Marie Curie* can be a *chemist* as well as an *experimental physicist*. While, in reality, some types are more plausible than others, we consider all types equally plausible due to a lack of evidence that would suggest otherwise. The ambiguity score $amb(i_{KB}, c_{KB})$ for an instance i_{KB} and a class c_{KB} is then calculated as the fraction of types of the instance that are a subclass of the class. For example, $amb(MarieCurie, physicist) = \frac{1}{2}$, since only one of the two types (i.e. chemist, experimental physicist) of the instance is a subclass of *physicist*. In contrast, $amb(MarieCurie, person) = 1$. The ambiguity score assigns a lower score, if there are more alternative class labels on the same level in the hierarchy and assigns the highest score ($= 1$), if there is only one potential class label at that level. It serves as a kind of *certainty* score.

Definition 2. (Specificity) The specificity of a class is determined by the number of instances that (directly or transitively) belong to this class. The score $spec(c_{KB})$ is calculated using Equation 1, where I is the total number of instances in the taxonomy and I_c is the number of instances of class c:

$$spec(c_{KB}) = \ln \frac{I}{I_c} \quad (1)$$

The equation is directly inspired by the notion of *idf (inverse document frequency)* in Information Retrieval, which assigns a higher score to terms that appear in only a few documents in a collection, compared to terms that appear in all documents. Similarly, we assign a higher score to classes that cover only a small subset of all instances and a lower score to classes that cover all instances. This score keeps generic class labels, such as *thing*, from receiving the highest scores, as we are more interested in specific labels.

Instance-level Classes In order to retrieve potential class labels for each distinct instance in the column, we take the following steps:

1. First, we identify instances in the taxonomy that match the instance in the column. In our column, we may find the instance term *A. Einstein*, while the taxonomy contains the term *Albert Einstein* (see Figure 3). We only consider the

Final edited form was published in "SAC 2015: Symposium on Applied Computing. Salamanca 2015", S. 1072–1077, ISBN 978-1-4503-3196-8
https://doi.org/10.1145/2695664.2695794

top n matching instances, using a word-by-word Levenshtein distance to compute the similarity $sim(i_{col}, i_{KB_j})$ between the instance terms.

2. For each matching instance, we then retrieve all classes the instance is a member of. We consider a class label, if there is a path of length $l \leq l_{max}$ between the instance and the class in the taxonomy.

3. For each retrieved class label, we compute a score (per matching instance) $s(i_{KB}, c_{KB})$ using Equation 2, which takes the ambiguity of the instance as well as the specificity of the class into account. Note that these scores can be precomputed once for the entire taxonomy, as both $amb()$ and $spec()$ are independent from the column instances.

$$s(i_{KB}, c_{KB}) = amb(i_{KB}, c_{KB}) \cdot spec(c_{KB}) \quad (2)$$

4. After scoring each class label, we retrieve the class label with the highest score $s(i_{col}, c_{KB})$, using Equation 3.

$$s(i_{col}, c_{KB}) = \max_{j \in \{1,\dots,n\}} (sim(i_{col}, i_{KB_j}) \cdot s(i_{KB_j}, c_{KB})) \quad (3)$$

Class Label Aggregation After retrieving scored class labels for each distinct instance in the column, we need to aggregate the scores to find the label that best describes the content of the entire column. The aggregation formula is shown in Equation 4, where I_{col} is the set of distinct instances in the column. The *top k* labels with the highest overall scores are then selected as suitable labels for the column. We can now use these labels to search for further column-specific information in the context, using the same matching approach as described in Section 3.

$$s_{col}(c_{KB}) = \frac{\sum_{i_{col} \in I_{col}} s(i_{col}, c_{KB})}{|I_{col}|} \quad (4)$$

4.2 Synonyms

Another limitation of the direct matching approach described in Section 3 becomes clear in the following example: considering the example in Figure 2c, we assume that, using an external knowledge base as described in the previous section, we obtain the label *garrison* for the entities in the first column. However, we can see that there are no mentions of the word *garrison* in the context. Instead, the word *fort* is used.
The direct string matching approach does not take into account that different terms may exist for the same concept. This is a common issue in many text search applications. Similar to query expansion in Web search, we address this issue by additionally searching for synonyms of the original search term. In our example, we can retrieve *fort* as a synonym of *garrison* using WordNet and then use the term *fort* to retrieve a number of relevant phrases from the context following the approach in Section 3.
We can apply synonym expansion to both, the original labels in the column header as well as the labels extracted from the knowledge base.

5. EXPERIMENTAL EVALUATION

To evaluate our approach, we performed a set of experiments on a corpus of Web tables. We present the results of a qualitative and quantitative analysis of the extracted noun phrases as well as our findings on the impact of this column-specific information on column and table search.

Context	# Tables	Avg. word count
Caption	43	4.72
Headline	406	2.13
Surrounding Text	770	228.03
Full Text	850	1,599.99

Table 1: Context statistics

Context	NPs
Caption	8
Headline	92
Surround. Text	1,513
Full Text	11,857

Table 2: Related NPs (per context type)

Algorithm	NPs
orig. label	3,457
orig. label syn.	1,173
yago label	4,442
yago label syn.	2,783
acronyms	2
all	11,857

Table 3: Related NPs (per extraction algorithm)

5.1 Data Set and Parameter Settings

For our experiments we extracted 850 tables containing non-informative column labels like *name* and *title* or generalized labels like *location* from English Wikipedia pages. For the inference of alternative column labels, we use the *Simple Taxonomy* in *YAGO* [10] as the external knowledge base. For the scoring functions (see Section 4.1), we set the parameters $l_{max} = 5$, $n = 1$ and $k = 3$. Furthermore, we used the Standford Parser [8] to identify noun phrases in the context and WordNet [5] for synonym expansion.

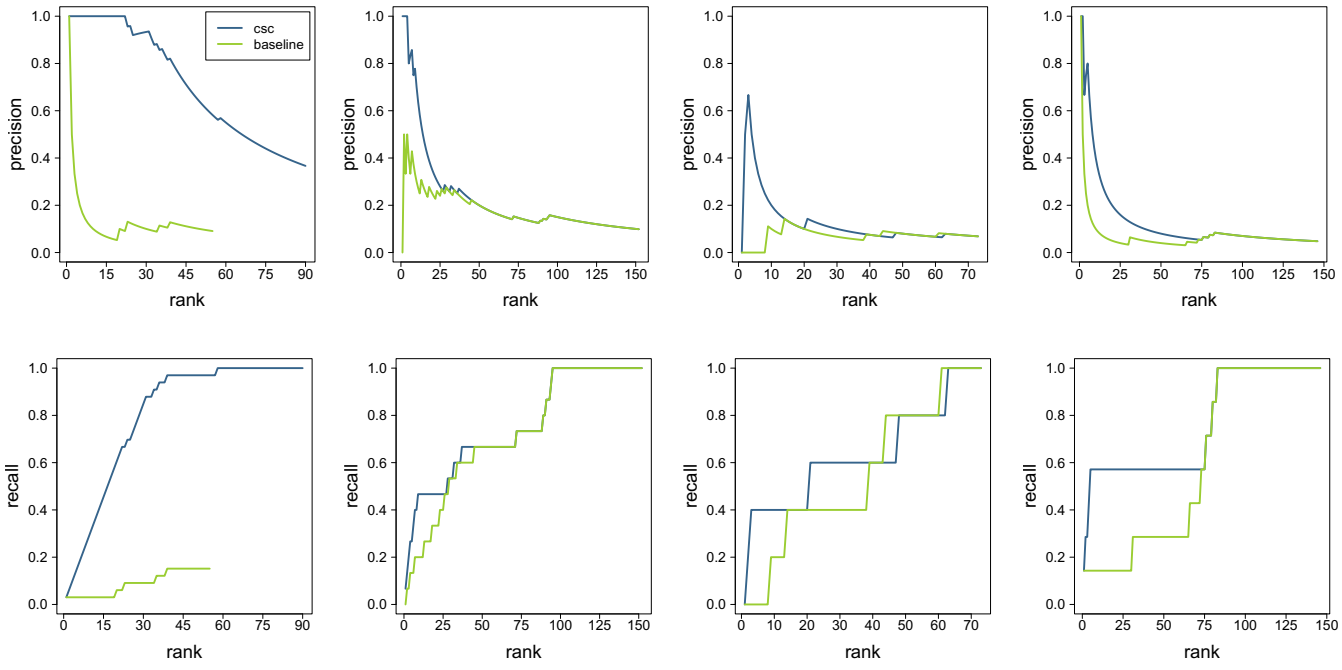
5.2 Annotation Quality and Statistics

To evaluate the accuracy of our approach, we extracted column-specific labels for all 850 tables. With an average of 4.15 columns per table, our algorithm extracted labels for an average of 2.23 columns per table. In total, we extracted 11,857 potentially related noun phrases from the context and 4,252 alternative column labels from YAGO. Tables 2 and 3 show the distribution of related NPs among the different context sections and extraction algorithms, respectively. In the latter, the *Algorithm* refers to the phrases that were used for matching, with *syn* indicating that synonyms of the respective labels were used.

As expected, most phrases where extracted from longer context sections, with only a few phrases extracted from the caption. However, only 5% of all tables had a *caption*-tag to begin with. Table 3 shows that most phrases were identified matching labels infered from YAGO to phrases in the context. This is most likely related to the fact that many of our tables contain labels like *name* and *title* that are less likely to appear directly in the context.

Annotation Quality For each of the different context sections as well as the extraction algorithms, we randomly selected 25 extracted phrases and manually evaluated their correctness. For each annotation we assigned a score of 1 if the phrase was a correct description of the column it was linked to, a score of 0.5 if the phrase was related but not an exact description, and 0 if the phrase was incorrect. Figure 5 shows the average score for each category.
The results indicate that the closer a phrase is to the table (on the web page), the more likely it is indeed related to a column in the table. While phrases in the *caption* and *headline* sections are of high quality, the overall label quality is much lower. Looking at the different extraction algorithms, we can see that matching based on synonyms is more likely to return incorrect phrases than directly matching original labels or YAGO labels.

Final edited form was published in "SAC 2015: Symposium on Applied Computing. Salamanca 2015", S. 1072–1077, ISBN 978-1-4503-3196-8
https://doi.org/10.1145/2695664.2695794



(a) football players (b) video games (c) short films (d) record label

Figure 4: Precision and Recall for column search queries

5.3 Search Quality

To evaluate the search quality, we used Lucene [2] to index all tables. For column search, we treated each column as an individual Lucene document and indexed, if available, the original column label, the column-specific phrases and the complete context, each in a separate text field. For table search, we used one document per table, indexing the complete header, all column-specific annotations and the complete context, again in separate fields. In both cases we assign boost factors of 2, 1.5 and 1 to the header, the annotations and the context, respectively. For the baseline, we only indexed the column or table header and the complete context, using the same boost factors.

We selected a set of phrase queries such that the search on the table headers alone would not return any results, which means that to identify relevant documents, additional context information was required. To measure precision and recall, we ran each query on all indexed data and manually labeled documents in the result sets as relevant or irrelevant.

Column Search For the column search we performed various phrase queries. Figure 4 illustrates precision and recall for the ranked results of the following four queries: *"football players"*,*"video games"*,*"short films"*,*"record label"*.

In the figures, *csc* stands for column-specific context. For each query, we can see that the column-specific annotations improve the ranking, resulting in higher precision and recall for the top-ranked columns. Query *"football players"* in Figure 4a shows the most significant improvement over the baseline. Here, more relevant columns have been identified using a combination of context and annotations than with only the context. This is due to the fact that, for many columns, we could infer the label *football players* directly from YAGO, while the context provided only very little information.

Table Search For the table search we used the same set of queries as for the column search, regarding a table as relevant, if at least one column in the table contained the desired entities. The results were very similar to the ones for column search. For lack of space, we only present results for two of the queries in Figure 6. Again, we can see that the column-specific annotations extracted by our algorithms lead to a higher rank for relevant tables.

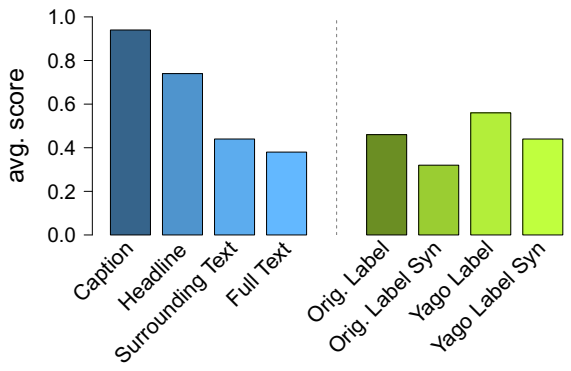


Figure 5: Avg. precision of related NPs

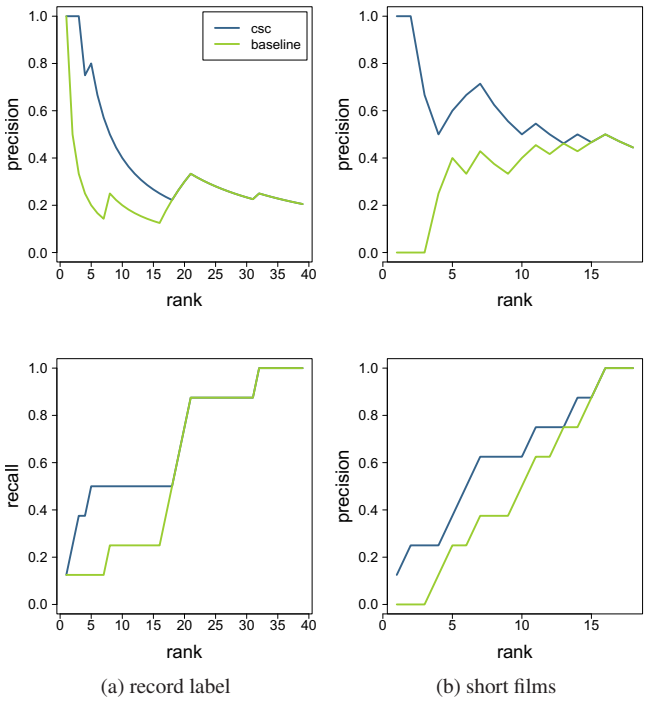
6. RELATED WORK

In recent years, research has focused on two information sources to improve the task of automatically understanding the content of a Web table: (1) the context of the table on the corresponding Web page, and (2) external common sense knowledge.

Web Table Context The text snippets surrounding the table on the Web page often contain relevant information to understand the content of the table. In most applications that make use of Web tables, the context is regarded as a bag of words that describes the

[2]http://lucene.apache.org/

Final edited form was published in "SAC 2015: Symposium on Applied Computing. Salamanca 2015", S. 1072–1077, ISBN 978-1-4503-3196-8
https://doi.org/10.1145/2695664.2695794



(a) record label (b) short films

Figure 6: Precision and Recall for table search queries

table as a whole instead of individual columns.
In [13], Yakout et al. use the surrounding text as one source to identify related tables in a large corpus. The similarity between two tables is computed using, amongst other scores, the tf-idf cosine similarity between the contexts and contents of the tables. In [3] and [7], the authors follow a similar approach, but instead of computing the similarity between tables, the goal here is to identify tables that are relevant for a search query. The tf-idf similarity between the query terms and the table's context and header is used to establish the relevance of the table. The authors give different scores to different context sections, but do not elaborate on the exact approach. Unlike our approach, all of these works do not extract column-specific information from the context.
The context has also been used by researchers to identify additional or *hidden* attributes of entities in the table. If all entities feature the same value for an attribute, it is very common that, instead of maintaining this column, the attribute is moved to and explained in the context. Identifying such *hidden* attributes in the context is the subject of the *CONTEXT*-operator proposed as part of the *Octopus*-system for web data integration in [1]. The authors propose different implementations of this operator, such as using tf-idf scores to identify significant terms in the context. In [4], Ling et al. address the same challenge in a more specific setting. For a set of tables with identical headers, extracted from related Web pages, the authors extract small text snippets from the context and align these across pages to identify hidden attributes. In contrast to extracting additional attributes from the context, our work focuses on extracting column-specific information for known attributes.

Annotating Web Tables The use of external sources of common sense knowledge, such as taxonomies and ontologies, for table understanding has been studied by many researchers in recent years. Publications differ mainly in the type of external knowledge source used as well as the specific labeling approach. In [3], Limaye et al. use complex graphical models for a holistic labeling approach based on YAGO. In [6], Mulwad et al. propose a very similar approach for joint inference based on a parameterized Markov network. Wang et al. leverage the probabilistic knowledge base Probase to label Web tables in [12]. First, they infer top-k class labels for each column and then select the most likely combination of labels across all columns based on confidence scores derived from Probase. In [11], Venetis et al. use custom is-A and relationship databases to derive column labels and relationship labels for column pairs, respectively. Based on scores in these databases, they apply a maximum likelihood model to infer the most likely labels. As far as we are aware, there has been no work so far that has linked the inferred labels back to the context of the table.

7. CONCLUSIONS

In this paper, we proposed a heuristic approach to extract column-specific context information for relational tables on the Web. We described different algorithms to extract directly as well as indirectly related phrases. Our results show that these column-specific annotations significantly improve ranking for table and column search compared to table-wide context information.
Future work on this topic includes further improving the quality of the extracted phrases using additional scores and supporting search queries other than entity name queries.

8. REFERENCES

[1] M. J. Cafarella, A. Halevy, and N. Khoussainova. Data integration for the relational web. *Proc. VLDB Endow.*, 2(1):1090–1101, 2009.
[2] J. Eberius, M. Thiele, K. Braunschweig, and W. Lehner. Drillbeyond: Enabling business analysts to explore the web of open data. *PVLDB*, 5(12):1978–1981, 2012.
[3] G. Limaye, S. Sarawagi, and S. Chakrabarti. Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.*, 3(1-2):1338–1347, 2010.
[4] X. Ling, A. Y. Halevy, F. Wu, and C. Yu. Synthesizing union tables from the web. In *IJCAI*, 2013.
[5] G. A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.
[6] V. Mulwad, T. Finin, and A. Joshi. Generating linked data by inferring the semantics of tables. In *VLDS*, pages 17–22, 2011.
[7] R. Pimplikar and S. Sarawagi. Answering table queries on the web using column keywords. *Proc. VLDB Endow.*, 5(10):908–919, 2012.
[8] R. Socher, J. Bauer, C. D. Manning, and A. Y. Ng. Parsing with compositional vector grammars. In *ACL '13*, 2013.
[9] S. Sorrentino, S. Bergamaschi, M. Gawinecki, and L. Po. Schema normalization for improving schema matching. In *ER '09*, pages 280–293, 2009.
[10] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A Core of Semantic Knowledge. In *WWW '07*, 2007.
[11] P. Venetis, A. Halevy, J. Madhavan, M. Paşca, W. Shen, F. Wu, G. Miao, and C. Wu. Recovering semantics of tables on the web. *Proc. VLDB Endow.*, 4(9):528–538, 2011.
[12] J. Wang, H. Wang, Z. Wang, and K. Q. Zhu. Understanding tables on the web. In *ER '12*, pages 141–155, 2012.
[13] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *SIGMOD '12*, pages 97–108, 2012.
[14] X. Yin, W. Tan, and C. Liu. Facto: a fact lookup engine based on web tables. In *WWW '11*, pages 507–516, 2011.