Julian Eberius, Patrick Damme, Katrin Braunschweig, Maik Thiele, Wolfgang Lehner

Publish-time data integration for open data platforms

Erstveröffentlichung in / First published in:

WOD '13: 2nd International Workshop on Open Data, Paris 03.06.2013. ACM Digital Library, Art. Nr. 1. ISBN 978-1-4503-2020-7

DOI: https://doi.org/10.1145/2500410.2500413

Final edited form was published in "WOD '13: 2nd International Workshop on Open Data. Paris 2013", Art. Nr. 1, ISBN 978-1-4503-2020-7
https://doi.org/10.1145/2500410.2500413

Publish-Time Data Integration for Open Data Platforms

Julian Eberius, Patrick Damme, Katrin Braunschweig, Maik Thiele and Wolfgang Lehner
Technische Universität Dresden
Faculty of Computer Science, Database Technology Group
01062 Dresden, Germany
firstname.lastname@tu-dresden.de

ABSTRACT

Platforms for publication and collaborative management of data, such as *Data.gov* or *Google Fusion Tables*, are a new trend on the web. They manage very large corpora of datasets, but often lack an integrated schema, ontology, or even just common publication standards. This results in inconsistent names for attributes of the same meaning, which constrains the discovery of relationships between datasets as well as their reusability. Existing data integration techniques focus on *reuse-time*, i.e., they are applied when a user wants to combine a specific set of datasets or integrate them with an existing database. In contrast, this paper investigates a novel method of data integration at *publish-time*, where the publisher is provided with suggestions on how to integrate the new dataset with the corpus as a whole, without resorting to a manually created mediated schema or ontology for the platform. We propose data-driven algorithms that propose alternative attribute names for a newly published dataset based on attribute- and instance statistics maintained on the corpus. We evaluate the proposed algorithms using real-world corpora based on the Open Data Platform opendata.socrata.com and relational data extracted from Wikipedia. We report on the system's response time, and on the results of an extensive crowdsourcing-based evaluation of the quality of the generated attribute names alternatives.

1. MOTIVATION

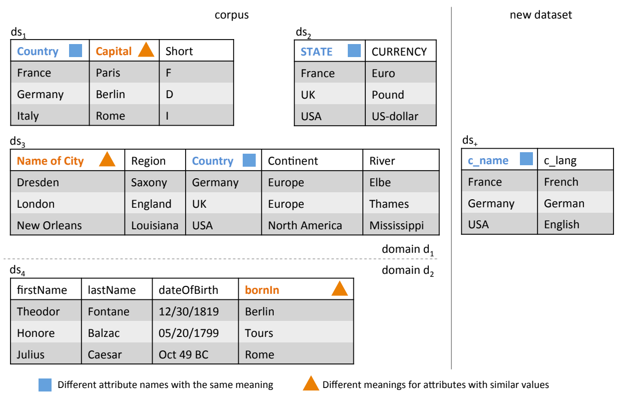
Platforms for collaborative collection and reuse of datasets are a current trend on the web. A prime example for platforms of this kind are Open Data Platforms, such as `data.gov` or `data.gov.uk`, where government agencies publish datasets of public interest. But this trend is not limited to governmental efforts: organizations, corporations and citizens have become data publishers and editors too. Another example for collaborative data management is Wikipedia, which contains over a million relational-style tables in its current English version. The free-for-all nature of these data publishing platforms leads to a strong heterogeneity in the corpora managed by these platforms, which contradicts the primary goals: data reusability and composability. One specific problem of this heterogeneity is the schema vocabulary, i.e., the terms used as attribute names of the datasets on the platform. Since the data is published by various authors using different terms for the same concepts, this leads to the classic data integration problem of finding equivalences between attributes in different datasets.

While the conventional techniques of schema matching are in principle applicable to all of these problems, they are usually designed to be used at *reuse-time*, i.e., when it is clear which datasets should be integrated and reused together. These techniques will not help to prevent the increase of heterogeneity in repositories, where a growing number of authors publish a rising number of datasets.

One solution would be to force newly published datasets to conform to a centralized schema or ontology, in which case schema matching techniques could be applied. While this may be a viable approach for some repositories, enforcing or even just creating an integrated schema will be unfeasible for multi-domain repositories with authors acting totally independent of each other. Additionally, this would limit the number and diversity of published datasets, since the publishing effort would increase.

So if we assume that the repository in question does not have an integrated schema or ontology, data publishers are free to choose arbitrary attribute names, leading to degenerated schemata and thus increasing integration effort at reuse time.

Publish-Time Data Integration (PTDI). The system presented in this paper is based on the idea that given the right tool-support, some lightweight integration work can easily be done at *publish-time*, to make the the new dataset fit into the existing repository. When the user publishes a new dataset the PTDI system augments the schema by alternative attribute names using statistics it maintains about the attributes and corresponding instances on the platform. To illustrate this process consider Figure 1, which depicts an exemplary corpus $\mathcal{C}$ consisting of four datasets ds_1 to ds_4 in two different domains. Furthermore, consider the new dataset ds_+ that is to be added to the corpus. The system should generate the output {c_name $\longmapsto$ (Country, STATE)}, as the latter two attribute names are used in the existing corpus for country names. Since "Country" appears two times (first column in ds_1 and third column in ds_3) and "STATE" only one time (first column in ds_2), "Country" is ranked before "STATE". For the second attribute name "c_lang" in ds_+ there is no recommendation,


corpus

ds_1

Country 🟦	Capital ▲	Short
France	Paris	F
Germany	Berlin	D
Italy	Rome	I

ds_2

STATE 🟦	CURRENCY
France	Euro
UK	Pound
USA	US-dollar

ds_3

Name of City ▲	Region	Country 🟦	Continent	River
Dresden	Saxony	Germany	Europe	Elbe
London	England	UK	Europe	Thames
New Orleans	Louisiana	USA	North America	Mississippi

domain d_1

domain d_2

ds_4

firstName	lastName	dateOfBirth	bornIn ▲
Theodor	Fontane	12/30/1819	Berlin
Honore	Balzac	05/20/1799	Tours
Julius	Caesar	Oct 49 BC	Rome

new dataset

ds_+

c_name 🟦	c_lang
France	French
Germany	German
USA	English

🟦 Different attribute names with the same meaning ▲ Different meanings for attributes with similar values

Figure 1: Example corpus $\mathcal{C}$ (4 datasets in 2 domains)

because no similar attribute exists in the corpus.
Notice however, that fitting names for an attribute are not only dependent on its value set, but also on the *domain* it stems from. This should be illustrated by an additional dataset ds_4 belonging to a different domain (see Figure 1). Similar to ds_1 it contains an attribute with city names, but the meaning of the attributes differ (*bornIn* versus *Capital* and *Name of City*). Therefore, in order to give correct name alternatives, our system also takes the existing domains on the platform and the domain of the new dataset into account. Specifically, it maps new datasets into one of the existing domains on the platform, before generating attribute alternatives based on statistics for this distinct domain.
Finally, in contrast to classic data integration techniques, *publish-time* data integration has much stronger performance constraints. The publish-time data integration algorithms have to be fast enough to allow instant publisher feedback.
The rest of this paper is organized as follows: we will propose four different algorithms for generating equivalent attribute names based on corpus statistics and discuss our domain classification method in Section 2. In Section 3 we will report on our crowdsourcing-based evaluation, in which we measured quantity and quality of the generated attribute name suggestions using real world corpora from `opendata.socrata.com` and Wikipedia. Finally, we will discuss related and future work in sections 4 and 5 respectively.

2. PUBLISH-TIME DATA INTEGRATION

Figure 2 gives an overview of the PTDI system's architecture. The system operates in an offline phase, in which corpus statistics are generated, and an online phase, in which these statistics are exploited to quickly generate attribute name equivalences for new datasets being added to the corpus.
In the *offline phase*, the datasets in the corpus are clustered into *domains*, which is necessary to deal with the different meanings of terms in different contexts, as explained in Section 1. Furthermore, the clustering into domains is leveraged to reduce the effort and time needed to compute equivalences in the online phase. Collecting domain-specific statistics allows us to map incoming dataset into one particular domain which is much smaller in size compared to the whole corpus. These domain-related processes are described in Section 2.1.
In the *online phase*, the attribute name equivalences for the incoming dataset are created. This phase consists of two steps. First, the already mentioned mapping into a specific domain and second, the comparison between values associated with the new attribute and the domain statistics. The specific implementations of this process discussed in Section 2.2 differ in the types of statistics they maintain, and the way the comparison between them and the new dataset is performed.

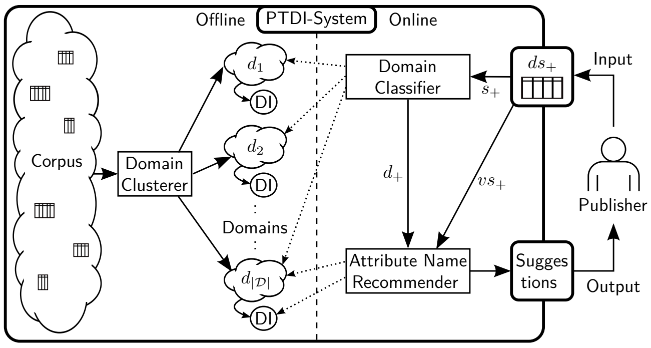


Figure 2: PTDI System Architecture

2.1 Domains and Domain-Classification

We refer to a *domain* as a set of datasets that have a common topic and thus use a common vocabulary. For the clustering of datasets into domains we follow Mahmoud and Aboulnaga [5]. Their approach clusters datasets based on their attribute names by creating feature vectors for each dataset that represent the vocabulary used in its attribute names and then performing bottom-up clustering using these vectors. The output of their algorithm is a set of triples, containing a dataset, a domain, and a probability of the given dataset belonging to the domain. Note that in our context, one dataset may belong to more than one domain, so domains may overlap.
To classify an incoming dataset ds_+ the same principles are applied. For ds_+ a feature vector must be created that represents the incoming dataset's vocabulary. Then, the most similar schema s_{sim} in the corpus is calculated by comparing the new feature vector to all existing vectors using a similarity function based on the Jaccard coefficient. If a schema with non-zero similarity is found, the algorithm returns the domain d_+ to which the dataset with the most similar schema s_{sim} belongs with the highest probability.

2.2 Equivalence Algorithms

Given the domain d_+ for a new dataset ds_+ we are now able to generate possible attribute equivalencies. To this end, we developed four algorithms that differ in the set of datasets they consider, and in the offline-generated statistics they use. Their input and output is defined in the same way: they assume a set of values vs_+, of the attribute k of the new dataset ds_+, and a domain d_+, and return a list of possible attribute equivalences to attributes existing on the platform. Note that all four implementations are only defined for text columns, i.e., columns that contain mostly string data.

2.2.1 Naive-$\mathcal{C}$ and Naive-d_+

The baseline approach Naive-$\mathcal{C}$ searches the complete corpus (not using d_+) for value sets that are similar to vs_+ and all attributes names belonging to these value sets are returned as possible equivalences. The similarity $vsSim(vs_x, vs_y)$

Final edited form was published in "WOD '13: 2nd International Workshop on Open Data. Paris 2013", Art. Nr. 1, ISBN 978-1-4503-2020-7
https://doi.org/10.1145/2500410.2500413

```
Input: Value Set $vs_+$
Output: Possible Equivalences
1 $names \leftarrow empty\ list$
2 foreach $vs_i \in \mathcal{VS}(\mathcal{C})$ do
3     if $vsSim(vs_+, vs_i) \geq th_{vsSim}$ then
4         $names \leftarrow names : a_i$
5 return $freqSort(names)$
```

Figure 3: The Naive-$\mathcal{C}$ approaches to equivalence generation

between two value sets vs_x and vs_y is calculated by counting the number of values v_x from vs_x for which a similar value v_y can be found in vs_y and the other way around. The sum of both is then normalized with the sum of the cardinalities of the two value sets which leads to similarity score in the interval $[0, 1]$. The similarity between two values is calculated using string distance measures, specifically the average between the Longest Common Substring-, Levenshtein- and Ngram-distance. The two values v_x and v_y are deemed similar if the value of this function is greater than a threshold. Based on this similarity measure the algorithm Naive-$\mathcal{C}$ is constructed as shown in Figure 3. In this approach, the input value set is globally compared to all value sets in the corpus, saving all the attribute names of the other value sets with sufficient similarity (see line 3). The resulting list of attribute names is ordered by their frequency in the corpus. As a first improvement to the baseline, instead of considering the whole corpus (see $\mathcal{VS}(\mathcal{C})$ in line 2 of Figure 3) Naive-d_+ only considers the value sets of the domain d_+ the incoming dataset was classified into it. Of course this approach requires a correct classification of the datasets as described in Section 2.1. Limiting the search for alternative attribute names to the domain has the obvious advantage that fewer value sets have to be compared, which increases performance. On the other hand, it can be argued that correct equivalences might be lost when considering only a subset of the whole corpus. However, as illustrated in Figure 1, similar value sets do not necessarily indicate similar meaning if they originate from different domains. For this reason we expect that Naive-d_+ will not only outperform Naive-$\mathcal{C}$ in run time, but also in quality of the generated equivalences.

The common disadvantage of both algorithms presented so far is that they perform all of their work *online* which contrasts with our requirement that publish-time data integration calls for fast response times, since data publishers are unlikely to cooperate if they suffer through long waits. Therefore, we developed the following two algorithms which use statistics created offline to improve the response time of the equivalence generation.

2.2.2 *Clustering*-d_+

The basic idea of the Clustering-d_+ algorithm is to add another clustering step on the domain level. In this offline step, similar value sets within each domain are clustered, and a list of possible attribute names for each of these clusters is saved. Using these clusters, the online step can be reduced to comparing the incoming value set to one representative value set of each of these domain level clusters leading to substantial performance gains. In the following, we will describe the offline and online phase of Clustering-d_+ in more detail.

Offline. The mentioned clustering of the value sets is performed as bottom-up clustering where initially every value set $\mathcal{VS}(d)$ in a domain is a cluster of it's own. Then the two most similar clusters are successively merged until there is no pair of clusters above a threshold. The similarity between two value sets is calculated using the function $vsSim$ from Section 2.2.1. For each of the clusters in the resulting set C a representative value set rvs is created. This representative value set is used in the online phase to reduce the number of comparison that have to be performed with the incoming dataset's values. It uses a dynamic threshold depending on the most frequent value to decide which values that appear in the value set cluster to include in the cluster's representative set. All the attribute names used for the value sets that were assigned to the same cluster are collected, ordered by frequency, and are saved as a list of possible equivalent attribute names. These representative value sets plus their respective list of attribute names constitute the domain statistics that are used by Clustering-d_+.

```
Input: Value Set $vs_+$
Input: Domain $d_+$
Output: Possible Equivalences
1 $(rvs_{max}, r_{max}) \leftarrow \arg\max_{(rvs,r) \in DI_{d_+}}$
2                                  $rvsSim(vs_+, rvs)$
3 if $rvsSim(vs_+, rvs_{max}) > 0$ then
4     return $r_{max}$
5 else
6     return $empty\ list$
```

Figure 4: Online phase of Clustering-d_+

Online. Leveraging these statistics the online phase of the Clustering-d_+ (see Figure 4) becomes more easier and more efficient. Each incoming value set vs_+ now has to be assigned to the most similar representative value sets rvs, which is already precomputed. In contrast to the similarity measure $vsSim$ from Section 2.2.1, we have to use $rvsSim$ which is only dependent on the number of values from rvs that have a partner in vs, but not the other way around. This is due to the construction of rvs which deliberately omits rare values, while the incoming value set vs may contain values of any frequency. Using the modified measure $rvsSim$ the most similar representative value set in the domain is identified, and the prepared list of possible equivalences is returned.

2.2.3 *Analysis*-d_+

The final algorithm, Analysis-d_+, creates different domain statistics compared to Clustering-d_+. Instead of clustering value sets and saving attribute names that belong to them, this algorithm groups value sets with the same attribute name. This leads to differences both in the form of the domain statistics and the way in which the online phase generates equivalences for the incoming dataset.

Offline. The offline phase of the algorithm is actually simpler than Clustering-d_+. Instead of clustering value sets, it simply groups value sets by their attribute names, and then creates a representative value set for each group. In contrast to the domain statistics of Clustering-d_+, which consist of tuples of representative value set with a corresponding list of attribute names, Analysis-d_+ creates triples consisting of representative value set, a single attribute name, and the frequency of this attribute name. Saving this frequency is necessary as the list of equivalences and its ranking is generated in the online phase of the algorithm, as opposed to

Final edited form was published in "WOD '13: 2nd International Workshop on Open Data. Paris 2013", Art. Nr. 1, ISBN 978-1-4503-2020-7
https://doi.org/10.1145/2500410.2500413

```
   **Input**: Value Set $vs_+$
   **Input**: Domain $d_+$
   **Output**: Possible Equivalences
1  $names \leftarrow empty\ list$
2  **foreach** $(rvs, a, z) \in DI_{d_+}$ **do**
3      **if** $rvsSim(vs_+, rvs) \geq th_{rvsSim}$ **then**
4          $names \leftarrow names : \underbrace{a : \cdots : a}_{z-times}$
5  **return** $freqSort(names)$
```

Figure 5: Analysis-d_+

offline with Clustering-d_+.

Online. In the online phase of Analysis-d_+, the incoming value set can again be compared to the representative value sets saved in the offline phase, instead of comparing it to all other value sets in the domain. In contrast to Clustering-d_+ however, each of these representative sets is only associated with one attribute name, not with a complete list of equivalences. For this reason, it is not enough to find the most similar triple in the domain statistics. Instead all triples where the similarity between rvs and the incoming vs is above a threshold have to be collected. The gathered triples can then be sorted by their frequency value, so that the attribute names that are most commonly used are ranked higher.

3. EXPERIMENTAL EVALUATION

We conducted an extensive experimental study to evaluate 1) the runtime behavior of the domain clustering and the equivalence algorithms, 2) the result quality for the four different approaches, and 3) the impact of the existing corpus size on the number and quality of generated attribute name alternatives.

3.1 Experimental Setup

For our evaluation we conducted various experiments on real-world data. First, we used `opendata.socrata.com`, an Open Data platform, which encompassed $16,591$ datasets when we extracted our test corpus. From the complete list of datasets, we discarded all tables with less than 3 columns or 2 rows, as well as all tables with a size of more than 100MB. For our second corpus, we extracted tables from English Wikipedia pages, using techniques similar to the ones presented in [1]. We extracted a set of $1.7m$ relational tables from Wikipedia's HTML and, again, discarded tables that were either too small or too large. From the remaining tables we randomly selected three sets of different sizes: a set of $5,000$ tables ($Wiki5k$), of $10,000$ tables ($Wiki10k$) and of $20,000$ tables ($Wiki20k$), with $Wiki5k \subset Wiki10k \subset Wiki20k$. These sets of different sizes enabled us to analyze the impact of the corpus size on the recommendation quality.

In addition to the corpora, we also extracted different classes of test data from both sources as input tables for the algorithms. For each potential test table, we counted the frequency with which the column name of each text attribute appeared in the respective corpus. We then used the highest encountered frequency to normalize the values and divided the resulting frequency range into six classes. The first class f_0 contains all attributes that appear in the test data, but not in the corpus (i.e. the frequency equals 0). Classes $f_1 - f_5$ are equal to the quintiles of the frequency distribution. Finally, we selected the tables so that the overall test

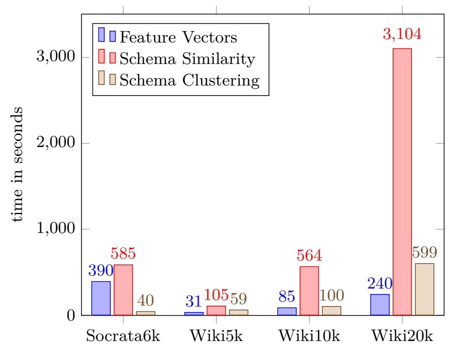


Figure 6: Runtime for individual processing steps and complete clustering process.

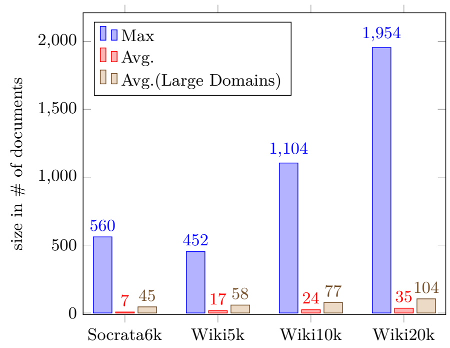


Figure 7: Max. and avg. # of documents in clustered domains (domains with a size > 10 count as large domains).

set contained at least 100 attributes for each frequency class. For all runtime experiments, we used a 64-bit Linux (Ubuntu 11.10) system with 8GB RAM and 2 Intel Xeon CPUs(each 2.4GHz). Additionally, we performed several experiments to evaluate the quality of the recommendations produced by the system. In general, an impartial quality assessment requires the evaluation of the results by actual users of the system. However, such an evaluation can be very extensive and time consuming when performed on a large scale. Therefore, we utilized the functionality and services provided by crowdsourcing platforms for the quality assessment.

3.2 Domain Clustering

At first, we performed the domain clustering as described in Section 2.1 for all test corpora.

Runtime Requirements Figure 6 shows the runtime required for all the individual processing steps performed during the domain clustering. The results show that computation of the similarities between the document schemas is the most expensive step dominating the other processing stages. This is due to the quadratic complexity of the pairwise schema matching performed in this step. Since the domain clustering is performed once offline before performing the equivalence algorithms a longer runtime for this step is not an immediate issue.

Concerning the extraction of the additional domain information required for Clustering-d_+ and Analysis-d_+, we found that this step requires significantly less time in Analysis-d_+ then with Clustering-d_+, where pairwise comparison of value set are necessary.

Domain Size For each corpus the domain clustering algorithm generates a large number of domains of different sizes. The maximum and average domain sizes are presented

Final edited form was published in "WOD '13: 2nd International Workshop on Open Data. Paris 2013", Art. Nr. 1, ISBN 978-1-4503-2020-7
https://doi.org/10.1145/2500410.2500413

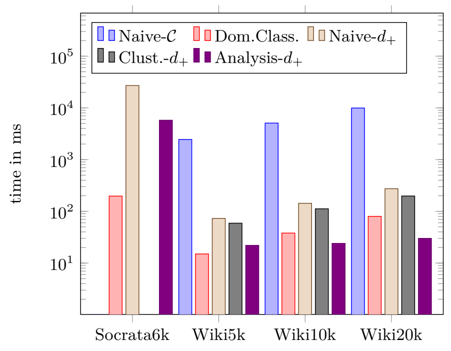


Figure 8: Average runtime for domain classification and attribute name recommendation (log scale).

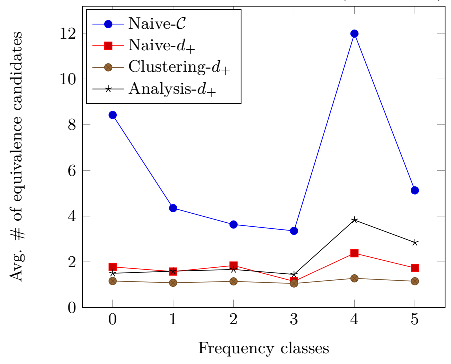


Figure 9: Avg. # of equivalence candidates found for input attributes using corpus $Wiki20k$, by frequency class.

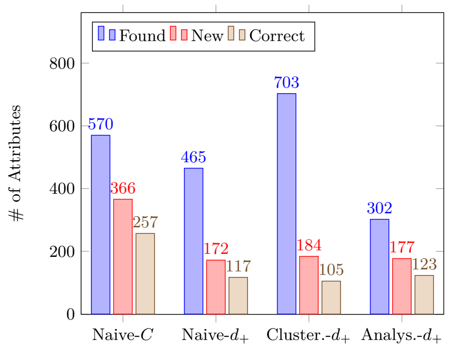


Figure 10: Overall, new and correct number of recommended attributes.

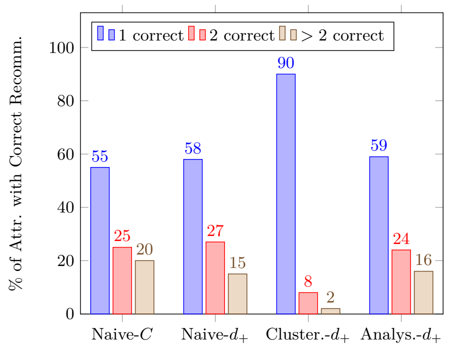


Figure 11: Percentage of attributes with exactly 1, exactly 2 or > 2 correct recommendations in the top 5.

in Figure 7. As expected, the average domain size increases with the corpus size, e.g., from 17 for the Wiki5k corpus to 35 for the Wiki20k corpus. The reason is that the number of domains is naturally bounded while the number of documents continuously increases with the corpus size. We further see that there are a few very large domains containing up to 1,954 documents for the Wiki20k corpus. However, there is also a long tail of domains which contain only a single document.

3.3 Equivalence Algorithms

Based on the domain clusters and domain information generated during the offline phase, we evaluated the different name equivalence approaches described in Section 2. For each test set (i.e. $Socrata$ and $Wikipedia$) we tested all four algorithms. For $Wikipedia$, we also varied the corpus size as described earlier. Figure 8 shows the average runtime per table. The time required for domain classification must be added to the runtime of all $* - d_+$ algorithms. While the $Wikipedia$ set shows acceptable runtime behaviour, the $Socrata$ set requires considerably more time for the recommendations. This is caused by the fact that $Socrata$ does contain some exceptionally large tables in the corpus. For two of our approaches, we even canceled the experiments due to these runtime issues. Overall, Analysis-d_+ is the fastest of the four algorithms, achieving a response time less than 100ms in most cases.
Figure 9 shows how many different equivalence candidates the respective algorithms were able to propose. As expected Naive-$\mathcal{C}$ generates much more diverse equivalences as it considers the whole corpus for each input set, while the other three algorithms just consider the one domain the input dataset is mapped into, which will contain a much smaller set of potentially equivalent names.

3.4 Recommendation Quality

In addition to the runtime experiments, we also analyzed the quality of the recommended equivalent attribute names. We leveraged the crowd-sourcing platform $CrowdFlower$ in order to have users evaluate the correctness of the recommendations. Before we ran the tasks on the crowd-sourcing platform, we took the top 5 recommendations for each attribute, for which the four algorithms generated results. If less than 5 attribute names were returned by an algorithm, we only used these. Since we were looking for meaningful alternatives to the original attribute name, we removed mentions of the original name from the top-5 set. Figure 10 shows the fraction of the recommendations which included new attribute names (i.e. not only the original attribute name). For this fraction of recommendations, we generated crowd-sourcing tasks. For each attribute name, we displayed the original table and highlighted the column in question. Below the table we displayed up to five alternative names recommended by our algorithms. We asked the crowd-workers to mark names which they thought were meaningful alternatives to the original name. Each individual task was given to three different crowd-workers and the answers were accepted if at least two of them agreed on them.
First of all, we analyzed the number of attributes for which at least one alternative name was marked as correct by the crowd. Figure 10 shows the results for all four approaches using $Wiki20k$ as the corpus. The dataseries "Found" describes the number of attributes for which recommendations could be found. "New" denotes the number of recommendations that were different from the original attribute name,

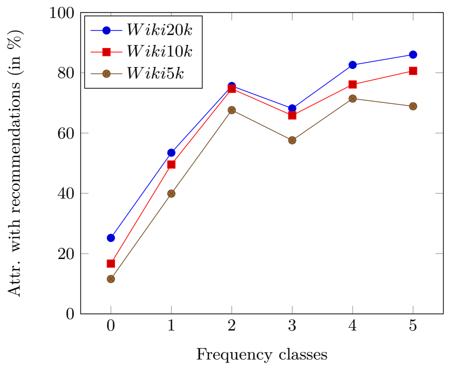


Figure 12: Fraction of attributes for which Clustering-d_+ was able to generate recommendations by frequency class.

whereas "Correct" counts the recommendations which have been verified positively by the crowd-workers. In most cases, correct recommendations were found for about 70% (e.g. 257 of 366 attribute names for Naive-$\mathcal{C}$) of the attributes evaluated through the crowd. Only Clustering-d_+ showed a slightly lower precision. Still, the majority of recommendations contained at least one meaningful equivalent.
Depending on the datasets available in the corpus, there can be more than one meaningful alternative in the top 5 recommendations. Therefore, we analyzed the crowd-sourcing results to see how many alternative names were found to be correct. Figure 11 shows the percentage of attribute names for which exactly one, exactly two or more than two correct alternatives were found in the top 5 recommendations. Again, Naive-$\mathcal{C}$, Naive-d_+ and Analysis$-d_+$ show very similar characteristics, with exactly one correct recommendation in $55-59\%$ of the cases. Clustering-d_+, on the other hand, has a significantly small fraction of attributes with more than one correct recommendation. This result is clearly caused by the fact that this approach generates significantly less recommendations for each attribute in the first place. Naive-$\mathcal{C}$ produces the largest amount of correct name alternatives, which is expected, since it uses the most comprehensive matching approach. The optimization techniques of the other algorithms clearly affect the result quantity.

3.5 Corpus Size

Finally, we also analyzed the impact of the corpus size on the recommendation quality. It is expected that a larger corpus is more likely to contain tables that are similar or related to the test table. Therefore, a larger corpus should enable more name recommendations. To test this hypothesis, we used the three *Wikipedia* corpora, which are subsets of each other. We formed domain clusters for each corpus and used these to generate recommendations for the same set of test tables. Figure 12 presents the results only for Clustering-d_+, but the remaining algorithms show a similar behavior. We can see that with an increasing corpus size for each frequency class more attribute names are generated. Using the results from the crowd-based evaluation, we can also show that a larger corpus does not only result on more recommendations, but in more correct ones.

4. RELATED WORK

Schema-matching approaches leveraging additional knowledge provided by a corpus share some basic concepts with our approach. Madhavan et al. [4] applied machine learning techniques on an existing corpus and known schema mappings in order to improve the number of matches. *Statistical schema matching* [3] tries to find a hidden probabilistic model within a set of input schemas. Further, the authors propose a subclass of their model, that handles the problem of synonym discovery. Chen et al. proposed the *Schemr* system [2] which is a tool for exploring and locating schemas or schema fragments in large schema collections in order to reuse them in another context. Therefore, the user has to provide an incomplete schema as well as optional search terms. By combining schema matching and text search techniques with a structurally-aware scoring metric Schemr is able to return a set of schemata that potentially match the input schema. Whereas Schemr is more a schema exploring tool serving different types of applications focus we focus on the specific problem of proposing single attributes during data publication.

5. CONCLUSIONS

Collaborative data management platforms such as Open Data Repositories are a new trend on the Web. As thousands of users continuously publish data to these platforms, the schema vocabulary steadily increases leading to highly heterogeneous corpora. In this paper, we therefore proposed the concept of data integration at *publish-time* where the attribute names of new dataset are aligned with the corpus to which the dataset should be added. We showed that Naive-C produces the largest amount of correct name alternatives but at the same time is also orders of magnitudes slower than the other three approaches. However, publish-time data integration algorithms must be fast enough to enable instant user feedback during the publishing process. Therefore, *Clustering-*d_+ and *Analysis-*d_+ provide a good mix between result quality and runtime. Using these algorithms we are able to provide attribute name alternatives during *publish-time* and thus to tame the growth of heterogeneity in collaborative data management platforms.

Acknowledgments.

This work has received funding under project number ESF-080951805 by means of the European Regional Development Fund (ERDF), the European Social Fund (ESF) and the German Free State of Saxony.

6. REFERENCES

[1] M. J. Cafarella, A. Y. Halevy, Y. Zhang, D. Z. Wang, and E. Wu. Uncovering the relational web. In *WebDB*, 2008.

[2] K. Chen, A. Kannan, J. Madhavan, and A. Halevy. Exploring schema repositories with schemr. *SIGMOD Rec.*, 40(1), July 2011.

[3] B. He and K. C.-C. Chang. Statistical schema matching across web query interfaces. SIGMOD '03, New York, NY, USA, 2003. ACM.

[4] J. Madhavan, P. Bernstein, A. Doan, and A. Halevy. Corpus-based schema matching. ICDE '05, 2005.

[5] H. A. Mahmoud and A. Aboulnaga. Schema clustering and retrieval for multi-domain pay-as-you-go data integration systems. SIGMOD '10, New York, NY, USA, 2010. ACM.