Generating Schema Labels through Dataset Content Analysis

Zhiyu Chen, Haiyan Jia, Jeff Heflin, and Brian D. Davison
Lehigh University
Bethlehem, PA
{zhc415|haiyan.jia|jeh3|davison}@lehigh.edu

ABSTRACT

Impoverished descriptions and convoluted schema labels are common challenges in data-centric tasks such as schema matching and data linking, especially when datasets can span domains. To address these issues, we consider the task of schema label generation. Typically, schema labels are created by dataset providers and are useful for users to understand a dataset. The motivation behind the task is that a lot of data linking systems require overlapping information between two datasets and rely on unique identifiers of schema labels. Moreover, it is common for schema labels in different datasets to have different identifiers even when they refer to the same concept. With no naming standard for schema labels, unintelligible labels are widely found in real-world datasets. For example, many schema labels contain abbreviations and compound nouns that hinder automated matching of attributes in corresponding datasets. Through schema label generation, more common (and thus understandable) schema labels can be provided to allow for broader schema matches in contexts such as dataset search and data linking. We develop a variety of features based on analysis of dataset content to enable machine learning methods to recommend useful labels. We test our approach on two real-world data collections and demonstrate that our method is able to outperform the alternative approach.

CCS CONCEPTS

• **Information systems** → **Information integration**; **Specialized information retrieval**;

KEYWORDS

data linking; text normalization

ACM Reference Format:
Zhiyu Chen, Haiyan Jia, Jeff Heflin, and Brian D. Davison. 2018. Generating Schema Labels through Dataset Content Analysis. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3184558.3191601

1 INTRODUCTION

Organizations and individuals worldwide make datasets public and enable users to freely explore such valuable resources. An increasing number of online data sources, such as governmental data portals (e.g., data.gov[1], data.gov.uk[2] and data.gov.sg[3]), and more general facilities like datahub[4] (alongside older sites such as the UCI machine learning repository[5]), suit the diverse needs of data experts, researchers, and journalists. More data also means more challenges. It becomes a non-trivial task to effectively integrate datasets from different resources. In order to help agencies to manage their data, the U.S. government released a policy[6] to instruct (government) data providers to provide metadata with their datasets. It is a great opportunity for technical communities to bring heterogeneous data together for diverse applications and also a big challenge which may require advanced approaches to manage the datasets. However, many datasets do not adopt metadata standards so that open source data management systems (e.g., CKAN[7]) are unable to be utilized. Another challenge is that different agencies are likely to have different data formats and standards [27] resulting difficulties in merging heterogeneous datasets.

Among all types of data, tabular data or the data table is one of the most important. It presents relational data in a compact way and is commonly used in different applications such as knowledge management and web data presentation. A data table usually has a header row, consisting of schema labels (attribute names), followed by data rows storing the actual data values of corresponding attributes. In this paper, we focus on this simple data table format although there are data tables with more complex structures where headers are nested. Tabular data is widely used in different communities because it clearly shows the relationships of different entities and facilitates data analysis. Many tools can easily work on tabular data for analysis and visualization.

Current data linking systems usually rely on the overlapping information in data itself or more commonly, the corresponding metadata fields such as title, tags, description and publisher. However, non-dictionary words (NDWs) commonly appear in data tables and can have a significant impact on data linking. The existence of NDWs in schema labels is also a well-known problem in schema matching systems [21, 24].

To address this problem, we propose a supervised method which recommends alternative schema labels. Considering Tables 1 and 2 from different domains, though we can easily identify that the column "latitude" in Table 1 refers to the same concept as "lat" in Table 2, it is not an easy task for data linking and schema matching systems. If we can recommend the column "lat" with new schema

WWW '18 Companion, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5640-4/18/04.
https://doi.org/10.1145/3184558.3191601

[1] https://www.data.gov/
[2] https://data.gov.uk/
[3] https://data.gov.sg/
[4] http://datahub.io/
[5] http://archive.ics.uci.edu/ml/index.php
[6] https://project-open-data.cio.gov/policy-memo/
[7] https://ckan.org

Table 1: Sample from a dataset of NYC Farmers Markets.

Farmers Market Name	...	State	Zip Code	Latitude	Longitude
Carroll Gardens Greenmarket	...	NY	11231	41	-74
Cortelyou Greenmarket	...	NY	11226	41	-74
Cypress Hills Youthmarket	...	NY	11208	41	-74
East New York Farm Stand	...	NY	11207	41	-74
East New York Farmers' Market	...	NY	11207	41	-74
Fort Greene Park Greenmarket	...	NY	11205	41	-74
...	...	...	...	...	...
Graham Avenue Farmers' Market	...	NY	11206	41	-74

Table 2: Sample from an Oceanographic dataset.

cruiseid	year	si	month_gmt	day_gmt	time_gmt	...	lat	lon
EN319	1999	T.Durbin	2	21	29.3	...	41.4922	-71.4187
EN323	1999	J.Ledwell	5	14	1146.88	...	41.5234	-70.6723
EN330	1999	C.Greene	10	23	140.4	...	42.5035	-66.8025
OC342	1999	B.Houghton	5	24	19.5	...	41.0683	-67.4617
...	...	...	...	...	...	...	...	...
OC343	1999	D.Hebert	6	25	731.47	...	40.9997	-67.6014

labels such as "latitude", "location", it can not only facilitate integrating the column of Table 2 with other columns, but also help users to better understand the meaning of this column.

We construct a variety of features from column content and enable machine learning models to generate alternative schema labels. To evaluate our method, we test on datasets with different heterogeneity and show that the features are effective for the schema label prediction task. Additionally, we experiment on integer columns, float columns, string columns and show that our method provides consistent performance on those different columns types.

We summarize our contributions as follows:

- We propose a domain-independent method for schema label prediction.
- We run experiments on real world datasets with varying degrees of heterogeneity and demonstrate the effectiveness of our methods on the task of schema label prediction. Our experimental results suggest that the difficulty of the task increases with the heterogeneity of the datasets.
- We evaluate our method on columns of three basic data types (integers, floats, and strings) and demonstrate that our method outperforms the baseline on each of them.

2 RELATED WORK

Though there is no prior work on the specific task of schema label prediction, work in the area of schema matching and data linking is related to our task and discussed below.

2.1 Schema Matching

In the database community, schema matching is a critical problem for integrating heterogeneous data sources, which aims to find pairwise-attribute correspondence in different schemas. It is similar to our task, except that they do not require that the pair of schema labels are exactly the same.

According to the classification of Rahm and Bernstein [20], there are two major types of schema matchers: schema-only matchers and instance-based matchers. Schema-only matchers are limited to schema information such as schema name, description and data type. For example, Sorrentino et al. [25] develop a lexical annotation technique to help identify similar schema labels. However, the result of lexical annotation is strongly affected by the presence of non-dictionary words in schema labels such as abbreviations and compound words. For this reason, they expand abbreviations with the help of an online dictionary and enrich WordNet with meanings of compound nouns. The output of their system can be used as the input of another schema matching system and improve the performance of schema matching. Ratinov and Gudes [21] solve the abbreviation issue by manually designing abbreviation patterns and reduce it to a supervised pattern classification problem. As we can see, the quality of schema labels have a huge impact on schema-only matchers and thus those methods put effort into the analysis of abbreviations and compound nouns in schema labels. Even for well-known schema-only matchers, such as Artemis [4], Cupid [14] and COMA [6], they require a specified external dictionary to measure the similarity of schema labels at some steps. In real world datasets, abbreviation and compound nouns cannot cover all the complex patterns of schema labels. Sometimes the column name of a data table has no real meaning or is even missing. Moreover, available schema information of real world datasets is limited.

Our method is closer to instance-based methods where we give insight into the data content. Since we train a supervised model to use a set of existing schema labels to annotate other schema labels, the results are less sensitive to NDWs. Besides, the similarity between two schema labels relies on the similarity of corresponding content rather than the surface form of the label text.

Automatch [2] uses machine learning techniques to automate the schema matching process. Their model acquires probabilistic knowledge stored in an attribute dictionary which characterizes different attributes by a set of possible values and their probability

estimates. Actually, this method is similar to our baseline method which characterizes a column by its bag-of-Words representation. As mentioned by the authors, there could be endless possible values for a column, especially for those whose data types are continuous variables. Therefore, instead of considering each column value as a feature, we consider each character of each column value as a feature. We also explore other higher level features from column content and better characterize different schema labels.

2.2 Data Linking

In recent decades, a large number of datasets have been published in different data repositories and it becomes an infeasible task to manually link different datasets. Under this context, data linking has become an important task which aims to automatically interlink datasets and facilitate their reuse.

Nikolov et al. [17, 18] present a keyword-based method with two main steps. In the first step, they use a subset of labels in the dataset as keywords to search for potentially relevant entities in external data sources. In the second step, they filter out irrelevant datasets by measuring semantic similarities used in ontology matching techniques. Leme et al. [11] propose a probabilistic method for Linked Data datasets. For a set of known datasets, they first construct a directed graph from the metadata to describe their connections. Then given a new dataset, they rank those datasets given a rank score function. A similar graph-based method is proposed by Lopes et al. [13] which treats dataset linking as a link prediction problem in social network. Ellefi et al. [1] propose a recommendation approach for data linking. They adopt the notion of dataset profiles, where a dataset is characterized as its textual descriptions and a set of schema labels. Therefore, given a source dataset, a cluster of comparable datasets can be retrieved based on their semantic similarities to a source dataset and each dataset can be ranked by tf-idf cosine similarity. A similar approach is proposed in [8], where a topic-dataset bipartite graph is produced during the topic modeling process; thus a dataset can be represented as a set of topics and a topic can be modeled as a set of significant datasets. Therefore a candidate dataset can be interlinked based on connectivity within the topic-profiles graph. In [5], the authors propose a user feedback-based approach to incrementally identify new datasets for domain-specific linked data applications. They first filter datasets according to the application queries and then use user feedback to analyze the relevance of candidate datasets.

As pointed by Nikolov et al. [17], finding the degree of overlap among datasets is critical for data linking. Schema label prediction can be a potential solution to increase the connectivity of heterogeneous datasets by recommending a dataset with schema labels that have appeared in other datasets.

2.3 Semantic Table Interpretation

As embedded data on web pages, Web tables take an important role in applications like knowledge base construction [22, 23, 32] and question answering [12, 19]. Therefore, it becomes crucial to recover semantics of Web tables.

There are three main tasks in semantic table interpretation [33]: 1) annotate columns in a table with semantic concepts; 2) identify the semantic relations between columns; and 3) cell disambiguation by linking them to entities in a knowledge base. Among the three tasks, the first task is the closest to our work. TableMiner [33] uses features from context inside and outside of the table to help annotate columns containing entity mentions. Venetis et al. [29] leverage a database to attach a class label to a column if a sufficient number of the values in the column are identified with the corresponding label in the database. Wang et al. [30] use Probase to annotate a Table with related concepts. Similarly, a large number of works [15, 16, 26] also make use of knowledge bases to interpret Web Tables.

Different from Web tables, real-world datasets usually have not enough context such as surrounding paragraphs or semantic markups inserted in the Webpage. Moreover, there are few entities that can be linked to a knowledge base since the concepts contained in a dataset are usually too narrow (e.g., street names on a map) or too broad. The method proposed in this paper only uses generic features extracted from the datasets and therefore only annotates columns with labels from the datasets rather than concepts from other resources.

3 PROBLEM STATEMENT

In this paper, we focus on finding alternative schema labels (column names) based on analysis of the content of the column.

We consider data tables with n columns and $m+1$ rows with the following format:

$$\begin{bmatrix} c_1 & c_2 & \dots & c_n \\ v_{1,1} & v_{1,2} & \dots & v_{1,n} \\ \dots & \dots & \dots & \dots \\ v_{m,1} & v_{m,2} & \dots & v_{m,n} \end{bmatrix}$$

For convenience, we use the following naming conventions in the rest of the paper:

- **schema label** (or column name): c_j, where $j \in [1, ..., n]$.
- **schema content** (or column content): $C_j = \{v_{1,j}, ..., v_{m,j}\}$, $j \in [1, ..., n]$
- **column**: $(c_j, C_j), j \in [1, ..., n]$.

Given C_j and k target labels $L = \{l_1, l_2, ..., l_k\}$, our objective is to learn a function that models $P(l|f(C_j)), (l \in L)$ where f is a function extracting features from C_j. The features will be introduced in the following section. A perfect prediction should satisfy:

$$c_j = \arg\max_{l \in L} P(l|f(C_j))$$

4 SCHEMA LABEL PREDICTION FEATURES

Our approach to predicting schema labels is to leverage features extracted from column content. Past research has indicated that useful features are important for table understanding [10, 31]. We assume that a schema label and evidence observed from the column content are highly related. One obvious example is that column contents corresponding to different data types are significantly different. Though column data types are not provided directly for the majority of public datasets, machine learning models are able to identify such features automatically [28].

Our task is much more challenging than just inferring the data type, since the number of data types is small and constant while the number of possible schema labels is uncertain but large. It also

means our model should be able to capture the differences among columns with the same data type. As shown in Table 1, a column of zip codes usually have data cells consisting of five digits, while a column of latitudes usually have data cells that are real numbers ranging from -90 to 90. If there is a column in the data table without a header and we know all the values in the column are five-digit numbers, the header is more likely to be "Zip Code" instead of "latitude". Therefore, for possible numerical columns, the *maximum value* and *minimal value* are important features to characterize them. However, not all columns are numerical columns. Then for non-numerical columns, we instead use the average maximum value and average minimal value of other columns in order to appropriately minimize the impact of these features.

We define *content unique ratio* and *content histogram* to describe the distribution of cell values. Content ratio [7] is usually used as a feature to categorize the class of table where the ratio of cells containing content of a specific type is calculated. Similarly, we use content unique ratio to categorize a column where the proportion of the number of unique cells over the number of all cells is calculated. In Table 1, the content unique ratio is $1/102 \approx 0.01$ for "State" if the table has 102 rows and all cell values under this schema label are all "NY". In contrast, the content unique ratio is $102/102 = 1$ for "Farmers Market Name" if all cell values under this schema label are different.

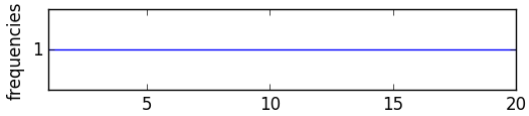
A content histogram contains more accurate information about the content distribution than the content unique ratio. To obtain the content histogram, we rank the unique cell values by frequencies (low-frequency first) and generate a vector where the i^{th} dimension is the frequency of the i^{th} ranked cell value. For different column contents, we could obtain the vectors of various lengths. For data.Gov and WikiTables which are two datasets used in our experiment, the medians are 26 and 13 respectively. Therefore, we generate the content histogram by resampling the vector to a 20-dimensional vector using FFT transformations[8]. We show the content histogram of "Farmers Market Name" and "Zip Code" of Table 1 in Figures 1a and 1b, respectively. The flatter shape of estimated frequencies of "Farmers Market Name" indicates the content distribution is closer to a uniform distribution than "Zip Code".

If we treat each column content as a document, then the schema label prediction can be seen as a document classification task, where classes are possible schema labels. So it is reasonable to incorporate bag-of-words (BoWs) representation as features. For column c, we construct the *BoWs features* as

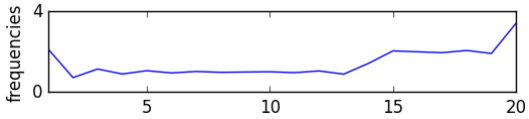
$$B_c = \{freq(u_1), ..., freq(u_i), ..., freq(u_n)\},$$

where n is the vocabulary size, u_i is the i^{th} word in the vocabulary and $freq$ represents the function calculating the frequency of u_i in c. To save memory, we only use character-level unigrams in BoWs (e.g., "EN319" is decomposed into "E","N","3","1" and "9"). In our experiment, we use BoWs features to construct the baseline method. The difference is that instead of considering character-level unigrams, we use TF-IDF representation of tokens extracted from column content.

[8]We use the method from https://docs.scipy.org/doc/scipy-0.17.0/reference/generated/scipy.signal.resample.html



(a) 20-dimensional content histogram of "Farmers Market Name" in Table 1.



(b) 20-dimensional content histogram of "Zip Code" in Table 1.

Figure 1: Examples of content histograms.

In a study of table header detection [9], Fang et al. showed that single row features could differentiate header rows and data rows. Inspired by their work, we extract the following *single column features* on each column instead of each row: number of characters, percentage of numeric characters, percentage of alphabetic characters, percentage of symbolic characters, percentage of numeric cells, average cell length, maximum cell length and minimum cell length. These features could be considered as an extension of the *BoWs features* which summarize the statistics of BoWs.

We summarize all the features in Table 3.

Table 3: A list of curated features for schema label prediction

ID	Feature length	Description
1	1	maximum value in the column content
2	1	minimum value in the column content
3	1	content unique ratio
4	20	content histogram
5	# of unique unigrams[9]	BoWs (character-level unigram)
6	1	number of characters
7	1	percentage of numeric characters
8	1	percentage of alphabetic characters
9	1	percentage of symbolic characters
10	1	percentage of numeric cells
11	1	average cell length
12	1	maximum cell length
13	1	minimum cell length

5 EXPERIMENTAL EVALUATION

In this section, we first discuss the datasets used in our experiments. Then we evaluate our method from two perspectives. In exact schema label prediction and normalized schema label prediction, we evaluate performance of the model and demonstrate the usefulness of the aforementioned features.

5.1 Datasets

For our first dataset, we collected all available comma-separated value (CSV) files (7485 in good format) from Data.gov which are

[9]741 for data.Gov and 54982 for WikiTables.

contributed by more than 50 U.S. government agencies. This dataset covers a variety of topics such as agriculture, climate, economy, and health. Web tables are also tabular tables and have an important role in applications like Web Data search and knowledge base construction. Therefore, we also experiment on WikiTables [3], which contains 1.6M tables extracted from Wikipedia.

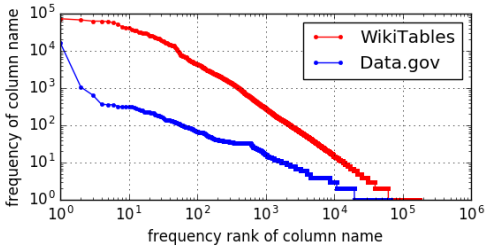


Figure 2: Rank-sorted frequencies of column labels

We observe that raw schema labels exhibit properties similar to terms in natural language, in that the rank-sorted frequencies of schema labels produce a curve which approximates the well-known Zipf's law and reflects the heterogeneity of schema labels.

5.2 Exact schema label prediction

We first evaluate our method on the task of exact schema label prediction. Considering a collection of tabular datasets where many schema labels are blank, our goal is to predict the missing schema labels by the corresponding column content. Specifically, we consider two questions: 1) How useful are features extracted from dataset content for schema label prediction? and 2) Does heterogeneity of the dataset collection make the task more difficult?

To answer the first question, we build machine learning models using the features proposed in Section 4, and evaluate the prediction results under different metrics. We treat schema label prediction as a multiclass classification task, where each schema label in the training set represents a class. We calculate macro-averaged and micro-averaged precision, recall and F-score of predictions on the test set. Macro-average is the mean of scores of all the classes, thus giving equal weight to each class. Micro-average, giving equal weight to each prediction decision, is the score obtained by globally counting the total true positives, false negatives and false positives. Larger classes have a larger contribution to the micro-average. In the multiclass classification scenario, the micro-average precision, recall and F-score are the same, thus we only show the Micro F-score in our results. We also report the top-n accuracy which is the fraction of test data for which the correct label is among the top-n labels considered most probable by the model.

The second question can be implicitly answered by comparing the results of the following datasets:

- **Gov_Rand**: 300 datasets are randomly selected from Data.gov.
- **Gov_NY**: 300 datasets are randomly selected from Data.gov published by NYC Open Data[10].
- **Wiki_Rand**: We experiment on 554218 tables from WikiTables which have at least 4 columns and 6 rows. Since a lot of tables are in unexpected format, we further filter those columns whose schema labels appear no more than 100 times.

The sizes of each dataset and training and testing partitions are found in Table 4.

Different data owners usually publish datasets in different domains, with different vocabularies and thus have different pattern of schema label creation. Thus, the difficulty caused by heterogeneity can also be shown by comparing the results on Gov_Rand and Gov_NY. Since there are only 327 datasets published by NYC Open Data, we randomly select 300 datasets for both Gov_Rand and Gov_NY so that the results from the models are more comparable.

Table 4: Statistics of extracted columns

Dataset	#train	#test	#classes
Gov_Rand	3833	1644	4048
Gov_Rand (freq >1)	1415	607	593
Gov_NY	2799	1200	2494
Gov_NY (freq >1)	1391	597	483
Wiki_Rand	806755	1882425	2234

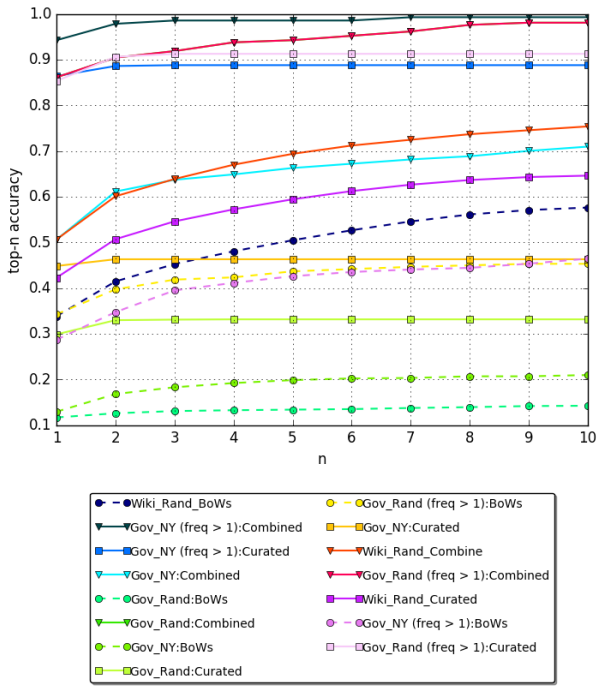


Figure 3: Top-n accuracy of exact schema label prediction

For our experiment, we train random forest classifiers using the curated features introduced in Section 4. The default parameters of scikit-learn implementation[11] are used except that the number of

[10] https://opendata.cityofnewyork.us/

[11] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Table 5: Micro average and Macro-average scores of exact schema label prediction

Features	Dataset	Micro-F	Macro-P	Macro-R	Macro-F
Curated	Wiki_Rand	0.42	0.30	0.21	0.23
	Gov_Rand (freq >1)	0.85	0.79	0.83	0.80
	Gov_Rand	0.30	0.13	0.15	0.14
	Gov_NY (freq >1)	0.86	0.79	0.83	0.80
	Gov_NY	0.45	0.18	0.21	0.19
BoWs	Wiki_Rand	0.34	0.34	0.16	0.19
	Gov_Rand (freq >1)	0.35	0.23	0.27	0.23
	Gov_Rand	0.11	0.05	0.06	0.05
	Gov_NY (freq >1)	0.28	0.12	0.14	0.12
	Gov_NY	0.13	0.03	0.04	0.03
Combined (Curated+ BoWs)	Wiki_Rand	0.43	0.30	0.22	0.24
	Gov_Rand (freq >1)	0.86	0.84	0.83	0.82
	Gov_Rand	0.37	0.24	0.25	0.24
	Gov_NY (freq >1)	0.94	0.82	0.85	0.83
	Gov_NY	0.49	0.31	0.32	0.30

trees in the forest is set up to 25 (to reduce memory requirements). As a **baseline**, we use the same classifier setting training on TF-IDF features extracted from column contents where each cell is tokenized.[12] It is important to notice that a large number of numeric values appear in the datasets which result in an extremely large vocabulary size. Thus dimension reduction is helpful in order to improve the classification efficiency. Truncated SVD[13] (a.k.a., LSA) is used to reduce the dimensionality of the TF-IDF representation and BoWs features to 300. We also concatenate the curated features with baseline BoWs features in order to see whether the combination of features could further improve the results. For Gov_Rand and Gov_NY, we split each of them into 70% training set and 30% testing set. Since the number of classes in Gov_Rand and Gov_NY is very large but the dataset size is relatively small, the results could be significantly affected by infrequent schema labels, especially those that only appear once. As a result, many labels in the testing set do not appear in the training set. Therefore, we also experiment on Gov_Rand and Gov_NY after filtering those columns whose schema label only appears once and we call them Gov_Rand(freq >1) and Gov_NY(freq >1) respectively.

Though the prediction must be wrong for the exact matching task, we still report the results of no filtering process and consider the evaluation of "wrong" predictions in the next section.

Experiment results are reported in Table 5. We observe that our curated features approach achieves better results than the baseline on all datasets. For both methods, scores on Gov_NY are higher than scores on Gov_Rand, which suggests the heterogeneity caused by data creators indeed increases the difficulty of the task. After filtering those schema labels that only appear once, the scores of both methods significantly increase, since the cases in which a class in the testing set does not appear in the training set have decreased. However, our method has a more considerable degree of improvement than the baseline and achieves an F-score greater than 0.8 for both the Macro-average and Micro-average. This indicates that our method has decent performance on popular schema labels. When applied to WikiTables, we notice that the gap of performance between our method and baseline is narrower. It is likely that schema label prediction is more like a text classification problem with regard to WikiTables. Compared to datasets on data.Gov, WikiTables have tabular data with smaller size, since tables with thousands of rows can hardly be displayed on a web page. A lot of datasets on data.gov are statistics and it is very likely that the content of single column is occupied by numbers. As content extracted from an encyclopedia, WikiTables have more text description about entities and thus the content of a column is closer to a document.

Figure 3 shows the top-n accuracy results. The performance on datasets from data.gov has no improvement when n is bigger than 3. As we discussed before, many of the labels in the testing set do not appear in the training set, and this sets an upper bound on the accuracy of exact schema label matching. For example, there are 594 columns whose labels only appear in the testing set of Gov_NY, therefore the accuracy can never exceed $(1200 - 594)/1200 = 50.5\%$. While for Wiki_Rand, the accuracy increases when n increases for both methods.

We calculated the gini importance of curated features of model trained on Gov_NY. For BoWs features and content histogram, we simply sum up the importance scores of all the dimensions. As a result, BoWs features and content histogram make the most contribution. If we consider each dimension as a single feature, then the most important three features are total number of characters, content unique ratio and the first dimension of content histogram.

From the above observations, we know that our method outperforms the baseline in all cases. Moreover, combining our method with baseline features can further improve the performance as expected. Since we only use character-level unigram features in our method and adding word-level TF-IDF features could relief this weakness. The prediction results can be significantly improved by filtering infrequent labels which means our methods can efficiently predict the schema labels especially for those popular ones. We also confirm that the difficulty of the task increases with the heterogeneity of the datasets.

[12] http://www.nltk.org/_modules/nltk/tokenize/toktok.html
[13] http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

5.3 Normalized schema label prediction

Exact schema label prediction is a pretty strict evaluation, since a true-positive requires the predicted schema label to perfectly match the original label of the tested column content. However, there are thousands of classes and the distribution is imbalanced as shown in Figure 2. It is possible that a class in the training set may not appear in the testing set. However, a "wrong" prediction could be potentially useful if it refers to the same concept. For example, consider the target label "Nationality": a semantically correct prediction from the model could be "Country". Therefore, we should not consider "Country" as a wrong prediction since they refer to the same concept. More examples are shown in Table 6.

Table 6: Examples of "wrong" predictions

Original labels	Predictions
Year	Season
Opponent	Team
Pos	Position
Score in the final	Score

In order to relieve the situation, we first do case-folding on schema labels and then rank them by their frequencies in Gov_Rand and Wiki_Rand. From the top 2000 schema labels, we normalize a label by another label in this set which is a synonym of the original label and more human readable. In addition, 89 labels annotated as uninterpretable are removed.

Similar to Section 5.2, we train separate models based on different features and datasets. The results of normalized schema label prediction on Gov_Rand and Wiki_Rand are reported in Figure 4 and Table 7. As expected, the scores under different metrics significantly increase. Besides, we notice that the top-n accuracy still grows when n is bigger than 3 on Gov_Rand, which is different from exact schema label prediction. It further indicates that our model can capture the relation between a schema label and its content.

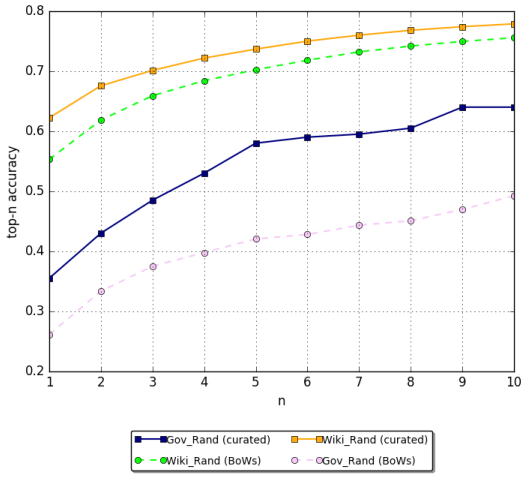


Figure 4: Top-n Accuracy of normalized schema label prediction

Table 7: Micro average and Macro-average scores of normalized schema label prediction

Features	Dataset	MicroF	MacroP	MacroR	MacroF
curated	Gov_Rand	0.36	0.21	0.22	0.20
	Wiki_Rand	0.62	0.33	0.29	0.30
BoWs	Gov_Rand	0.25	0.16	0.17	0.15
	Wiki_Rand	0.55	0.29	0.20	0.23

5.4 Evaluation on different data types

In this section, we evaluate schema label prediction methods on columns with different data types. We use pandas[14] to automatically infer the data type of a column and only keep those columns whose data types can successfully be identified by the IO tool. 1000 columns are randomly selected for integer type, float type and string type respectively. For each type of column, we train a model on 70% of data and evaluate on the rest of data. The experiment results for our method and baseline are reported in Table 8. As expected, our method outperforms the baseline on all the three types of columns. It is also interesting to notice that both methods perform the best on float columns while perform the worst on string columns. There could be two reasons. First,string columns have more unique labels than float columns and integer columns, which means predicting schema labels of string columns is inherently a more difficult task.Second, some of the features are based on the numeric values in the column content, while for string columns, such features are treated as missing values and calculated from average values from other columns. Such features could be useless for string columns and damage the performance of the model. This fact indicates that designing different features for different data types could further improve the performance of schema label prediction.

Table 8: Micro average and Macro-average scores on different data types

Features	Data type	MicroF	MacroP	MacroR	MacroF
curated	integer	0.31	0.11	0.12	0.11
	float	0.37	0.10	0.11	0.10
	string	0.23	0.11	0.12	0.10
BoWs	integer	0.25	0.10	0.11	0.10
	float	0.32	0.07	0.09	0.07
	string	0.20	0.08	0.09	0.08

6 CONCLUSION AND FUTURE WORK

We have considered the problem of schema label prediction based on the content of a column. We treat it as a multi-class classification task, in which each class represents a schema label. A variety of features are developed to solve the problem. Our method has been evaluated on two real-world datasets: tabular data collected from data.gov and WikiTables extracted from Wikipedia. We first evaluate the approach on exact schema label prediction, which requires the predicted label to exactly match the original schema label. In this task, our method clearly outperforms the baseline on

[14]https://pandas.pydata.org/

all datasets. We find that heterogeneity of the datasets likely makes the task more difficult.

Since the distribution of schema labels is quite imbalanced, many labels used in testing do not appear in the training set, which makes it inherently difficult to perform well. Therefore, we also experiment on top-ranked normalized schema labels. We select the most frequent 2000 schema labels across both datasets and merge labels that are synonyms. As expected, the scores under different metrics significantly increase compared with the results of exact schema label prediction. Additionally, we demonstrate that our method outperforms the baseline on columns of different data types. We notice that both methods perform the best on float columns while the worst on string columns, since some proposed features could be useless for string columns. It reminds us that using different features for different data types are necessary for schema label prediction. We will leave data type inference and schema label prediction for columns of different data types as future work.

One limitation of our current method is that we only consider the features from a single column, without considering its relationship with other columns co-occurring in the data table. Intuitively, if two columns are similar, then our method may give them the same schema label. However, it is unlikely for two identical columns to appear in the same data table. By considering the occurrence of other schema labels, such cases could be disambiguated.

One application of our method is to facilitate dataset retrieval. An existing challenge of dataset retrieval is that user queries seldom contain terms that are broadly used in schema labels, which results in low recall of related datasets. In our experiment, we find that our method often gives a prediction that are synonyms of original schema label or share the hypernym with the original schema label. Usually, the composition of NDWs schema labels is irregular and complicated but their synonyms or hyponyms could be searched by users. For example, for a column whose schema label is "Pos", our method could predict the schema label as "Position". However, "Position" is a term more preferred by users and more valuable to be indexed. We expect that using the predicted labels as possible term expansions (either for queries or at indexing time), the dataset retrieval system can have improved recall.

Acknowledgments

This material is based upon work supported by a Lehigh University internal Collaborative Research Opportunity grant.

REFERENCES

[1] Mohamed Ben Ellefi, Zohra Bellahsene, Stefan Dietze, and Konstantin Todorov. 2016. Dataset Recommendation for Data Linking: An Intensional Approach. In *The Semantic Web. Latest Advances and New Domains*. Springer, 36–51.

[2] Jacob Berlin and Amihai Motro. 2002. Database Schema Matching Using Machine Learning with Feature Selection. In *Advanced Information Systems Engineering*. Springer, 452–466.

[3] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. TabEL: Entity Linking in Web Tables. In *The Semantic Web - ISWC 2015*. Springer, 425–441.

[4] Silvana Castano and Valeria De Antonellis. 2001. Global viewing of heterogeneous data sources. *IEEE Trans. on Knowledge and Data Eng.* 13, 2 (2001), 277–297.

[5] Hélio Rodrigues de Oliveira, Alberto Trindade Tavares, and Bernadette Farias Lóscio. 2012. Feedback-based Data Set Recommendation for Building Linked Data Applications. In *Proc. 8th Int'l Conf. on Semantic Systems (I-SEMANTICS '12)*. ACM, 49–55.

[6] Hong-Hai Do and Erhard Rahm. 2002. COMA-a system for flexible combination of schema matching approaches. In *Proc. 28th Int'l Conf. on Very Large Databases*. Elsevier, 610–621.

[7] Julian Eberius, Katrin Braunschweig, Markus Hentsch, Maik Thiele, Ahmad Ahmadov, and Wolfgang Lehner. 2015. Building the resden web table corpus: A classification approach. In *IEEE/ACM 2nd Int'l Symp. on Big Data Computing (BDC)*. 41–50.

[8] Mohamed Ben Ellefi, Zohra Bellahsene, Stefan Dietze, and Konstantin Todorov. 2016. Beyond established knowledge graphs-recommending web datasets for data linking. In *Int'l Conf. on Web Engineering*. Springer, 262–279.

[9] Jing Fang, Prasenjit Mitra, Zhi Tang, and C. Lee Giles. 2012. Table Header Detection and Classification. In *Proc. 26th AAAI Conf. on Artificial Intelligence (AAAI'12)*. AAAI Press, 599–605.

[10] Matthew Hurst. 2001. Layout and Language: Challenges for Table Understanding on the Web. In *Proc. Int'l Workshop on Web Document Analysis*. 27–30.

[11] Luiz André P Paes Leme, Giseli Rabello Lopes, Bernardo Pereira Nunes, Marco Antonio Casanova, and Stefan Dietze. 2013. Identifying candidate datasets for data interlinking. In *Int'l Conf. on Web Engineering*. Springer, 354–366.

[12] Jimmy Lin and Boris Katz. 2003. Question Answering from the Web Using Knowledge Annotation and Knowledge Mining Techniques. In *Proc. 12th Int'l Conf. on Information and Knowledge Management (CIKM '03)*. ACM, 116–123. https://doi.org/10.1145/956863.956886

[13] Giseli Rabello Lopes, Luiz André P Paes Leme, Bernardo Pereira Nunes, and Marco A Casanova. 2014. RecLAK: Analysis and Recommendation of Interlinking Datasets.. In *LAK Workshops*.

[14] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. 2001. Generic Schema Matching with Cupid. In *Proc. 27th Int'l Conf. on Very Large Data Bases (VLDB '01)*. Morgan Kaufmann, 49–58.

[15] Varish Mulwad, Tim Finin, and Anupam Joshi. 2013. Semantic Message Passing for Generating Linked Data from Tables. In *The Semantic Web – ISWC 2013*. Springer, 363–378.

[16] Varish Mulwad, Tim Finin, Zareen Syed, and Anupam Joshi. 2010. T2LD: Interpreting and Representing Tables as Linked Data. In *9th Int'l Semantic Web Conf. (ISWC)*. 25.

[17] Andriy Nikolov and Mathieu d'Aquin. 2011. Identifying relevant sources for data linking using a semantic web index. In *WWW2011 Workshop: Linked Data on the Web (LDOW 2011)*.

[18] Andriy Nikolov, Mathieu d'Aquin, and Enrico Motta. 2012. What Should I Link to? Identifying Relevant Sources and Classes for Data Linking. In *The Semantic Web*. Springer, 284–299.

[19] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering Table Queries on the Web Using Column Keywords. *Proc. VLDB Endow.* 5, 10 (June 2012), 908–919.

[20] Erhard Rahm and Philip A. Bernstein. 2001. A survey of approaches to automatic schema matching. *The VLDB Journal* 10, 4 (01 Dec 2001), 334–350.

[21] L Ratinov and Ehud Gudes. 2004. Abbreviation expansion in schema matching and web integration. In *Proc. IEEE/WIC/ACM Int'l Conf. on Web Intelligence*. 485–489.

[22] Dominique Ritze, Oliver Lehmberg, Yaser Oulabi, and Christian Bizer. 2016. Profiling the Potential of Web Tables for Augmenting Cross-domain Knowledge Bases. In *Proc. 25th Int'l Conf. on World Wide Web (WWW '16)*. 251–261. https://doi.org/10.1145/2872427.2883017

[23] Yoones A Sekhavat, Francesco Di Paolo, Denilson Barbosa, and Paolo Merialdo. 2014. Knowledge Base Augmentation using Tabular Data.. In *LDOW*.

[24] Serena Sorrentino, Sonia Bergamaschi, and Maciej Gawinecki. 2011. NORMS: an automatic tool to perform schema label normalization. In *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*. IEEE, 1344–1347.

[25] Serena Sorrentino, Sonia Bergamaschi, Maciej Gawinecki, and Laura Po. 2009. Schema Normalization for Improving Schema Matching. In *Conceptual Modeling - ER 2009*. Springer, 280–293.

[26] Zareen Syed, Tim Finin, Varish Mulwad, and Anupam Joshi. 2010. Exploiting a web of semantic data for interpreting tables. In *Proc. 2nd Web Science Conf.*, Vol. 5.

[27] Barbara Ubaldi. 2013. Open government data: Towards empirical analysis of open government data initiatives. *OECD Working Papers on Public Governance* 22 (2013), 0_1.

[28] Isabel Valera and Zoubin Ghahramani. 2017. Automatic Discovery of the Statistical Types of Variables in a Dataset. In *Proc. 34th Int'l Conf. on Machine Learning (Proceedings of Machine Learning Research)*, Vol. 70. PMLR, 3521–3529.

[29] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Pașca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering Semantics of Tables on the Web. *Proc. VLDB Endow.* 4, 9 (June 2011), 528–538.

[30] Jingjing Wang, Haixun Wang, Zhongyuan Wang, and Kenny Q. Zhu. 2012. Understanding Tables on the Web. In *Conceptual Modeling*. Springer, 141–155.

[31] Yalin Wang and Jianying Hu. 2002. A Machine Learning Based Approach for Table Detection on the Web. In *Proc. 11th Int'l Conf. on World Wide Web (WWW '02)*. ACM, 242–250.

[32] Xiaolu Zhang, Yueguo Chen, Jinchuan Chen, Xiaoyong Du, and Lei Zou. 2013. Mapping Entity-Attribute Web Tables to Web-Scale Knowledge Bases. In *Database Systems for Advanced Applications*. Springer, 108–122.

[33] Ziqi Zhang. 2014. Towards efficient and effective semantic table interpretation. In *International Semantic Web Conference*. Springer, 487–502.